ESCUELA DE POSTGRADO
UNIVERSIDAD TECNOLÓGICA DEL PERÚ

TRABAJO DE INVESTIGACIÓN DE MEJORES PRÁCTICAS EN DESARROLLO DE SISTEMAS WEB CONTRA ATAQUE DE INYECCIÓN

PRESENTADO POR:

Juan José, Romero Aliaga

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRIA EN INGENIERÍA DE SEGURIDAD INFORMÁTICA

ASESOR: Carlos Daniel Rodríguez Vilcarromero

LIMA –PERU

2019

DEDICATORIA

A mis queridos hijos Juan José y Luciana por ser mi inspiración y estímulo para lograr mis objetivos en la vida.

A mi compañera de toda la vida Elsa por su amor, apoyo y paciencia.

A mis queridos padres Juan de Dios y Felicia ejemplos a seguir para mí.

AGRADECIMIENTO

Agradezco a mis docentes de la maestría por esos años de aprendizajes.

Así como también, agradezco a mi asesor Carlos Rodríguez por su apoyo adicional en la comprensión de las técnicas de investigación.

RESUMEN

El manejo de información en un tiempo real es lo que las empresas anhelan en la actualidad; para ello es necesario el uso de sistemas de información en sus diversas presentaciones. Pero al construir las aplicaciones descuidan la parte seguridad por el afán de obtener la información en la brevedad posible y los controles de seguridad no son implementados o son configurados de forma errónea y no son evaluados en su integridad; por ello los sistemas son vulnerables a diversos tipos de ataque por terceros, ocasionando una desconfianza en los usuarios y grandes pérdidas económicas para la organización.

El área responsable de la seguridad informática de las empresas suele preocuparse en la funcionalidad del sistema y no evalúa la calidad y seguridad; debido al costo alto que ocasiona en una empresa la validación, verificación e implementación de controles de seguridad. Gran parte de las vulnerabilidades que presentan se dan por errores de programación, falta de aseguramiento de control y calidad de software; que al ser alojados a un hosting están propensos a todo tipo de amenazas con consecuencias del alto riesgo.

En la investigación se realiza el estudio del ataque más frecuente en estos últimos veinte años (según estadísticas de ESET, Acunetix, OWASP entre otras instituciones de renombre) que es Inyección; se propone la manera correcta y segura para desarrollar aplicaciones Web contra ataques de Inyección. Porque al final estos errores básicos y/o comunes de programación se transforman en grandes problemas de seguridad, generando gran impacto económico en contra de la organización.

El estudio incluye una serie de lineamientos de las mejores prácticas de OWASP, SANS y/o CVE, componentes del estándar ISO 27001, COBIT, modelos de calidad de software y entre otros que asegure la protección contra ataque de Inyección.

Al término del estudio, se habrá obtenido:

- Recomendaciones de herramientas y/o técnicas para prueba de software contra ataque de inyección.
- Enunciados de buenas prácticas de codificaciones de software y uso de herramientas para evitar ataque de inyección.
- Recomendaciones de herramientas para testing de portales web contra ataque de inyección.

ABSTRACT

Today the development of web applications are being threatened by techniques and injection attack tools that are manually and automated.

The objective of this research is to show the best practices of web application development against injection attacks based on standards and software quality assurance models.

The study includes a series of guidelines of the best practices of OWASP, WASC, SANS and / or CVE, components of the ISO 27001 standard, COBIT, software quality models and among others that ensures protection against Injection attack.

In addition, it shows the tools and techniques to evidence, exploit and evaluate the vulnerabilities of the system.

Keywords: good practice, injection

ACRÓNIMOS

CMMI	Capability Maturity Model Integration
COBIT	Objetivos de Control para Información y Tecnologías Relacionadas
CVE	Common Vulnerabilities and Exposures
INEI	Instituto Nacional de Estadísticas e Informática
ISO	Organización Internacional para la Estandarización
ITIL	Information Technology Infrastructure Library
MPS	Mejora de Proceso de Software
OWASP	Open Web Application Security Project
PSP	Personal Software Process
SANS	SysAdmin Audit, Networking and Security Institute
SGSI	Sistema de Gestión de Seguridad de la Información
SQA	Aseguramiento de Calidad de Software
TMMI	Test Maturity Model Integration
TSP	Team Software Process
WASC	Western Association of Schools and Colleges

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Situación Problemática

Las Soluciones Informáticas para todo tipo de organización en estos últimos años es ya una necesidad. Parte de los procesos de la empresa se realiza mediante aplicaciones informática y la mayoría de ellas se realizan mediante un entorno web. Negocios como entidades bancarias, instituciones educativas, comercio electrónico, gubernamentales, redes sociales entre otros, procesan información mediante estos medios; transformándose como un activo principal en la toma de decisiones.

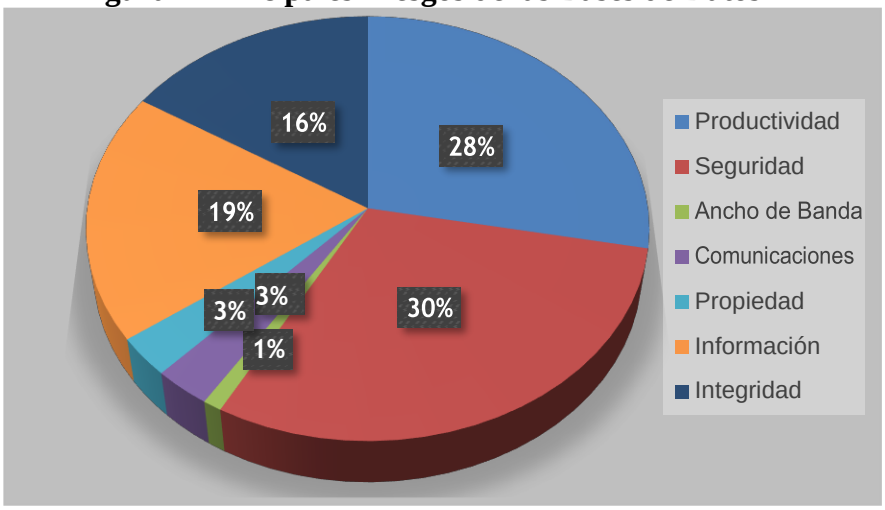
Además, las técnicas de ataque para acceder a equipos de Tecnología y conocimientos de Hacking por parte de los intrusos son accesible toda esta información en Internet u otro tipo fuente; lo que con lleva que las organizaciones deberían concientizar y mejorar sus medidas de protección en la implementación de sus Sistemas y Redes a estas posibles amenazas que perjudiquen los 3 pilares de la seguridad de información (integridad, confidencialidad y disponibilidad).

Teniendo en cuenta que no existen aplicaciones web 100% seguras, el rol que se debe asumir con respecto a seguridad de desarrollo de software es poder aminorar las vulnerabilidades y garantizar la continuidad del negocio mediante una planificación e implementación de controles de seguridad. Todo ello se puede cerciorar si las organizaciones establecieran un SGSI; que controle, evalúe y audite la aplicación durante su desarrollo y antes de ser implementada.

Según los altos porcentajes de ataques a portales web que presentan vulnerabilidades ocasionadas por errores de desarrollo de software; se puede afirmar que los encargados de la construcción de la aplicación carecen o desconocen sobre implementación de controles de seguridad en construcción de aplicaciones web para lograr desarrollar software de calidad. Influye esto de manera negativa en las empresas que construyen su propio sistema o es más crítico cuando son terceros (outsourcing) los que desarrollan el sistema.

Cuando una organización decide contar con un Sistema Web; pero no ha diseñado o no cumple con un plan de aseguramiento de calidad de software. Pone en riesgo los puntos que muestra la figura 1.

Figura 1 Principales Riesgos de las Bases de Datos



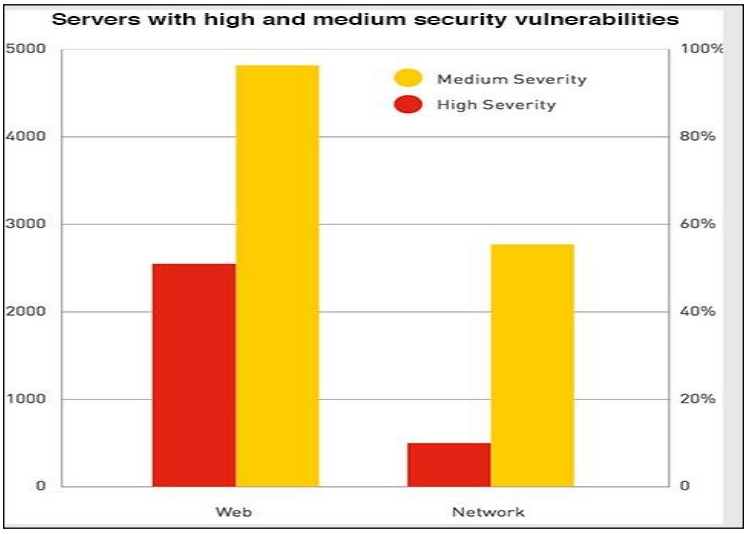
Recuperado de http://securecomputing.intelsecurity.com/

En la figura1, presenta un alto porcentaje de riesgo para las empresas con respecto a Información como activo principal, Seguridad y Productividad como continuidad de negocio ante un ataque a su sistema de información. Las posibles consecuencias podrían ser manipulación de la información, pérdida económica y desconfianza de los usuarios y futuros clientes.

La cantidad de ataques a soluciones web cada año se va incrementando debido a la facilidad de adquirir herramientas y conocimientos de técnicas de hacking, donde la manipulación y/o ejecución de estos no requiere de conocimiento avanzando. Pero las vulnerabilidades con respecto al desarrollo de la aplicación permanentemente vienen siendo lo mismo, a pesar de las recomendaciones y estadísticas frecuentes publicados por OWASP, SANS, CVE,ESET entre otros.

Acunetix informa que, en el año 2015, se han realizado pruebas de seguridad informática a 5500 organizaciones con un total de 15000 sitios web y redes además de 1,9 millones de archivos; en un porcentaje mayor a la mitad de los portales Web presenta vulnerabilidad a posible ataque de SQL Inyección o XSS.

Figura 2 Alarmantes estadísticas de sitios web con graves vulnerabilidades



Recuperado de https://www.helpnetsecurity.com/2015/07/17/nearly-all-websites-have-serious-security-vulnerabilities/

Según la figura 2, afirmo que la mayor parte de los ataques por errores de programación en una aplicación web es SQL Inyección, porque no implementaron controles que prohíban la ejecución de procesos que no están permitidos realizar. Los ataques se pueden realizar de forma manual o automatizadas haciendo uso de herramientas open source que permita al atacante acceder al sistema y obtener información y privilegios sobre la base de datos.

Las opciones de las técnicas de ataque SQL Inyección para ser ejecutados no requiere de nivel avanzado para visualizar la vulnerabilidad y explotar; a causa de las malas prácticas de programación por parte de los desarrolladores de software o en la etapa de implementación; pero si diremos que las consecuencias pueden ser de diferentes grados de riesgo, según el nivel de acceso y permiso otorgado al usuario.

En la tabla 1 mencionamos algunos casos relevantes a nivel mundial sobre ataque de Inyección de los últimos años.

Tabla 1 Casos Inyección SQL

Noticia	Fecha
Joven de 11 años en menos de 10 minutos ingreso a la web del estado de Florida utilizando SQL Lite; accediendo a información confidencial, que estaba almacenada en texto plano sin cifrar.	10/08/ 2018
Ataque de Inyección SQL deja al descubierto datos personales de 3000 afiliados al portal de Juventudes Socialistas de España.	18/04/2017
'Hackers' acceden al servidor web de CNI que almacena todos los archivos y contenidos de la página. Ejecutando Inyección SQL.	24/10/2017
Usan inyecciones SQL en ataques de Portales Web SEO según Akamai (División de Investigación de Amenazas)	20/01/2016
Ataque SQL inyección en sistema de censo universitario de INEI	09/10/2010
Hackean Portal Web del partido político PPC del Perú, usando técnicas de inyección.	28/09/2010

Los resultados estadísticos y los casos mencionadas fue el punto de partida para desarrollar el siguiente estudio; y mencionar el rol que deben de cumplir los encargados de desarrollar proyectos web al seleccionar modelo de calidad de software, buenas prácticas de programación, recursos y/o instrumentos para prevenir estos tipos de ataque de Inyección. El contenido de esta documentación maneja un lenguaje común y sencillo de comprender, con la única finalidad de crear conciencia en la construcción y evaluación de aplicaciones web.

1.2 Formulación del Problema

1.2.1 Formulación de Problema general

¿De qué manera los involucrados en desarrollar soluciones informáticas deberán construir Sistemas Web para evitar o contrarrestar los ataques de Inyección?

1.2.2 Formulación de Problemas específicos

1. ¿Cuáles son las técnicas y/o herramientas que identifique las vulnerabilidades contra ataque de Inyección a Soluciones Web?
2. ¿Qué herramientas usar para explotar y evaluar soluciones web contra ataques de Inyección?
3. ¿Cuáles son los errores típicos que no se debe realizar durante el desarrollo e implementación de aplicaciones web, que al ser ejecutados son propensos a los ataques de Inyección?

1.3 Objetivos de la Investigación

1.3.1 Objetivo General

Proponer a los involucrados en construir soluciones informáticas, las mejores prácticas de desarrollo contra ataques de Inyección a Sistemas Web .

1.3.2 Objetivos específicos

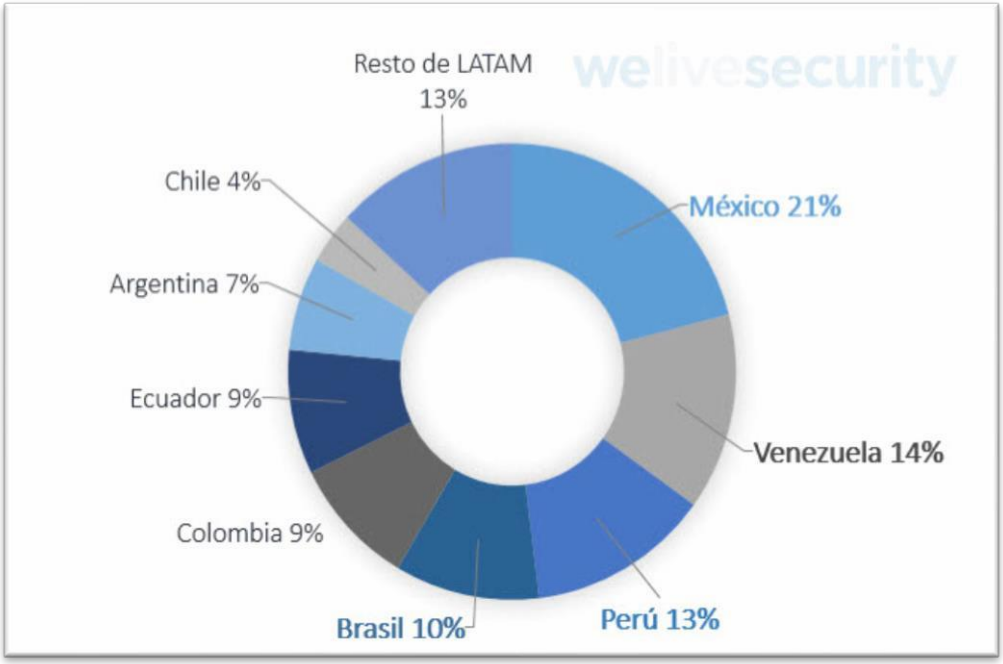
1. Identificar las zonas vulnerables de una aplicación web propenso a los ataques de Inyección a través de técnicas y/o herramientas de forma manual y automatizada.
2. Determinar las herramientas adecuadas para la explotación y evaluación de sistemas web contra ataques de Inyección.
3. Mostrar los lineamientos de seguridad a cumplir para evitar los errores típicos propensos a los ataques de Inyección durante el desarrollo e implementación de aplicaciones web.

1.4. Justificación de la Investigación

Desarrollar e implementar una aplicación web no solo es evaluar que funcione correctamente el proceso de datos según los requerimientos solicitados por el usuario, también implica realizar prueba de seguridad durante el proceso de solicitud de alguna información. Mucho de los portales web con que interactuamos permanentemente no han incluido el proceso de validación y verificación en seguridad de cada fase de desarrollo de software. Realizar pruebas de seguridad reduce los riesgos a consecuencia de ataques de terceros, disminuye las posibles vulnerabilidades de la web.

Según reporte anual de la entidad especializada en seguridad informática ESET en el año 2018, se reportó un máximo histórico en cantidad de vulnerabilidades en portales web. A continuación, la figura 3, muestra los porcentajes de detección de vulnerabilidades y ataque por país en Latinoamérica.

Figura 3 Porcentaje por país del total de víctimas del Cibercrimen

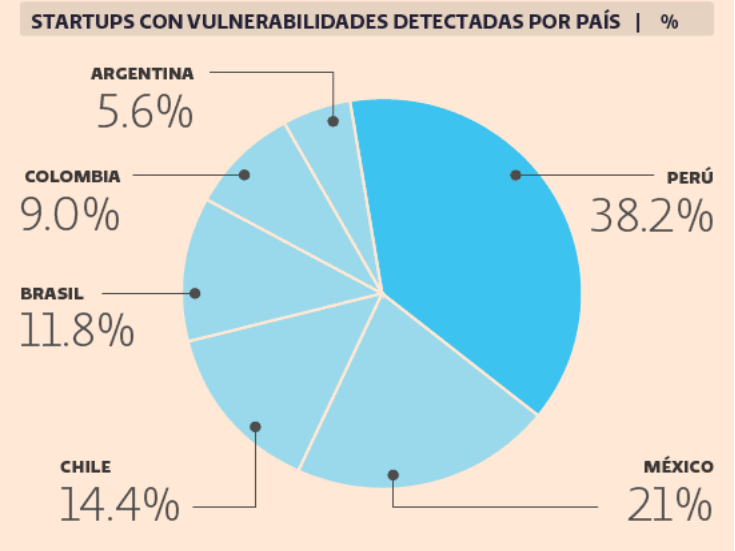


Recuperado de https://www.welivesecurity.com/la-es/2019/01/10/amenazas-informaticas-mas-afectaron-paises-america-latina/

Observamos que Perú ocupa un tercer lugar (13%) dentro de América Latina. Las vulnerabilidades se van incrementando cada año; para ello va a ser importante prevenir y concientizar en adquirir conocimientos sobre seguridad para poder implementarlos y aminorar el porcentaje de ataque.

Según el director de Boxug, en el año 2017; 1302 hackers presentaron 751 reportes de vulnerabilidades a empresas latinoamericanas. La estadística fue la que muestra la figura 4, Perú está ubicado en un primer puesto con un 38.2%.

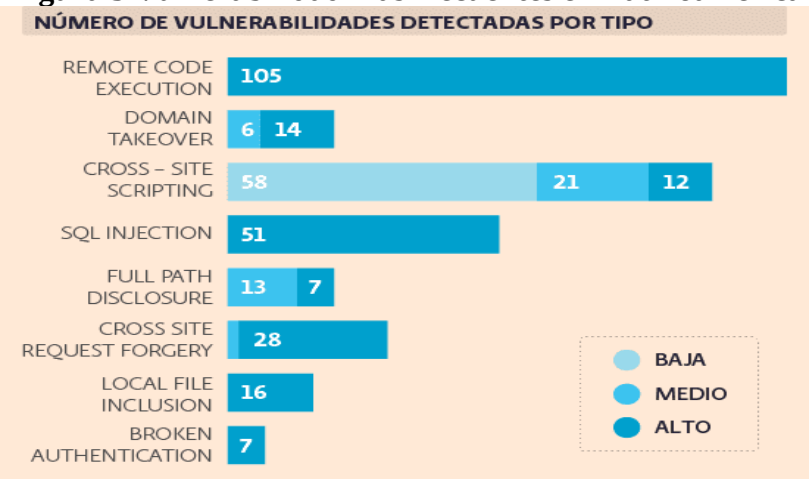
Figura 4 Países con Startups más vulnerables en Latinoamérica



Recuperado de https://www.eleconomista.com.mx/tecnologia/Boxug-combate-los-ciberdelitos-contra-las-startups-de-America-Latina-20171127-0090.html

De los reportes de las vulnerabilidades el mayor porcentaje corresponden a ejecución remota de código, scripting entre sitios y SQL Inyección. Boxug informa que estas vulnerabilidades son por errores de programación de los desarrolladores; siendo vulnerabilidades de alto riesgo y de fácil ejecución. Tal como muestra la figura 5.

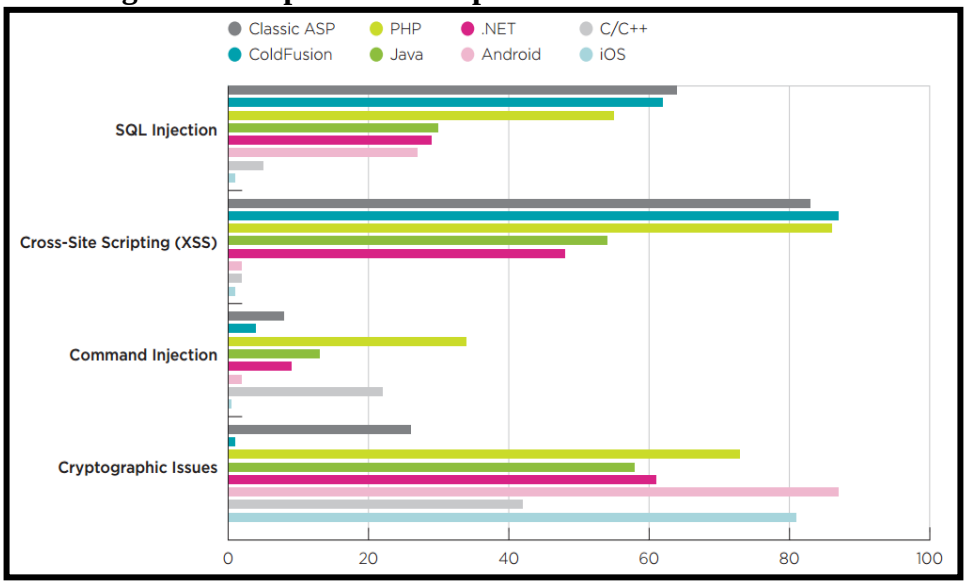
Figura 5 Vulnerabilidad más frecuentes en Latinoamérica



Recuperado de https://www.eleconomista.com.mx/tecnologia/Boxug-combate-los-ciberdelitos-contra-las-startups-de-America-Latina-20171127-0090.html

La figura 6 muestra las cuatro categorías principales de vulnerabilidad son: criptográfica, inyección SQL, Cross-Site Scripting e inyección de comandos. Se compara porcentaje de ataques más comunes vs el lenguaje de programación para desarrollar portales Web.

Figura 6 Comparación de tipos de vulnerabilidad crítica



Recuperado de **https://news.softpedia.com/news/top-programming-languages-that-generate-software-vulnerabilities-497101.shtml#sgal_0**

Se observa que las vulnerabilidades web como inyección SQL y Cross-Site Scripting son sustancialmente más prevalentes en aplicaciones escritas en lenguajes de script web como ASP clásico, ColdFusion y PHP, en comparación con las aplicaciones .NET y Java. Esto es muy probablemente debido a las diferencias en los conjuntos de características de cada idioma. Hay menos API de seguridad integradas en Classic ASP, PHP y ColdFusion que se han previsto para .NET y Java. A modo de ejemplo, hasta hace poco era muy difícil de obligar a los desarrolladores realizar consultas con parámetros desde un lenguaje de programación a una base de datos. Ahora existen una infinidad de framework a implementar a los lenguajes de programación, que para el atacante es más difícil de escribir código de inyección a una aplicación web. Todo ello resultara si el desarrollador conoce y cumple con la estructura de programación que sugiere el framework.

Según las estadísticas mostradas los ataques de inyección están afectando no solo a páginas del exterior también a las que son de Perú; entonces podemos afirmar que las soluciones web están siendo desarrollados por personas o entidades que desconocen o no cumplen correctamente la aplicación de controles de seguridad contra este tipo de ataque. Por lo tanto, el presente estudio justifica realizar una investigación que ayude a los involucrados a desarrollar soluciones web a implementar, explotar y evaluar controles de seguridad contra amenazas de Inyección. Los beneficios por obtener será el aseguramiento de calidad de soluciones web contra Inyección considerando:

- Las buenas prácticas desarrollo de soluciones web.
- Estándares Internacionales para desarrollo de software de calidad .
- Modelos Internacionales para mejora de calidad de software.
- Evaluación de la Solución Web.
- Implementación de recursos contra ataque de inyección.

1.5 Factibilidad del estudio

Como se mencionó anteriormente gran parte de los desarrolladores de soluciones informáticas web en Perú no acostumbra a realizar pruebas de seguridad antes de su implementación y alojamiento en internet solo realizan pruebas de buen funcionamiento de los procesos e ingreso/salida de información de la web; exponiéndose a posibles amenazas por terceros que puede afectar la continuidad y credibilidad de la institución.

El contenido de la investigación puede ser usado como referencia de buenas prácticas programación contra ataque Inyección, donde se indica el ¿Qué se debe tomar en cuenta? mas no ¿Cómo hacerlo? porque ello va a depender de los recursos y/o necesidades que pueda tener la organización o el equipo de desarrollo.

1.5.1 Factibilidad Técnica

Requerimientos de Hardware:

Como la investigación está relacionado con las buenas prácticas, lo que corresponde a requerimientos de hardware va a depender el tipo de portal web a desarrollar según la dimensión de información a manejar (micro, mediana o macro).

Lo que se sugiere es que los equipos (Servidores, WAF, IDS, IPS entre otros) con respecto a la seguridad perimetral sean de marcas líderes en seguridad y calidad en Informática; por lo tanto, al elegir se prioriza la

seguridad antes de lo económico. Y el buen funcionamiento de estos va a depender de la correcta configuración de los controles de seguridad.

Requerimientos de Software.

Existen diferentes lenguajes de programación para desarrollar soluciones web como: los lenguajes líderes en seguridad como Java, PHP y Phyton entre otros o los líderes en el mercado C#, Java entre otros. Se puede seleccionar cualquiera de los lenguajes mencionados, pero no solo depende de ello, si se desea obtener buenos resultados dependerá mucho del tipo de modelo de programación a desarrollar, las bibliotecas o librerías a implementar, las buenas prácticas de programación, entre otros para obtener una página robusta, segura y de buena calidad.

En requerimientos de software dependerá de la elección del uso de herramientas con licencia o gratuitas para el desarrollo de software, por ello es variables el costo. Para evaluación de software usaremos herramientas recomendadas por OWASP, aplicaciones Linux entre otras aplicaciones Open Source seguras.

1.5.2 Factibilidad Operativa

Para determinar la calidad de aplicación web se debe de elegir y cumplir un patrón de desarrollo de software según política de la organización basados en estándares Internacionales de Desarrollo de Software de Calidad tales como:

- ISO 12207

- Métrica
- ISO 9126
- ISO 15504

También los modelos Internacionales de Mejora de la Calidad de Software:

- Modelo TMMI
- Modelo TSP y PSP
- Modelo Six Sigma
- Modelo Moprosoft
- Modelo MPS
- Modelo CMMI

1.5.3 Factibilidad Económica

Como se indicó para el desarrollo de software dependerá del uso de herramientas a usar ya sea con licencia o de distribución libre; tanto en seleccionar el lenguaje de programación, base de datos, framework, librerías entre otros.

En la presente investigación con respecto a la prueba y testing del software se propone software gratuito y comercial. Con respecto al económico va a depender de la organización.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes de la investigación

2.1.1 Antecedentes Internacionales

(Amado Ballen , Ayala Calderón, & Sierra Morales , 2017), presentaron la tesis de grado **"Análisis de vulnerabilidades en aplicaciones Web desarrollados en PHP Versión 5.6.24 con base de datos MYSQL Versión 5.0.11 a partir de ataques de SQL Inyección",** oriento su investigación en la evaluación final de Sistemas Web desarrollados en PHP haciendo uso de la herramienta OWASP Zed Attack Proxy Proyect. Donde la herramienta realiza un análisis exhaustivo a la Web arrojando como resultado un reporte sobre los puntos vulnerables ataques de SQL Inyección, los grados de nivel de riesgo y las alternativas de solución. Propone como una de las alternativas de solución para estos tipos de ataque el uso de framework, implementación de la herramienta de defensa IDS (Detección de Intrusos del Sistema)
,validación de archivos con origen JavaScript sea de fuentes de confianza y establecer el encabezado Content-Type a través del servidor Web.

(BRIONES PINCAY & HERNANDEZ PEÑAHERRERA , 2018), presentaron la tesis de **"Auditoria de Seguridad del Servidor Web de la empresa Publynext S.A utilizando mecanismos basado en OWASP"**, menciona el rol que debe asumir una empresa cuando va a auditar un Sistema Web antes de la etapa ejecución. Para ello toma en cuenta las métricas que recomienda OWASP con respecto a las técnicas, herramientas y lineamientos a seguir. Con los resultados obtenidos realizar un plan de contingencia. Recomienda que las empresas deben de

realizar plan de pruebas en ambientes de trabajo local, asegurar una configuración correcta de los servidores, uso de recursos Open Source y establecer una política de seguridad en la empresa.

(Fuyo Gonzales & Rivera Oviedo, 2016), presentaron la tesis de **"Seguridad de JAX-RS frente a ataques por inyección de código",** afirman que en la actualidad no existe herramienta alguna que evalué las vulnerabilidades a Web Services Rest implementados con JAX-RS por ello desarrollan un prototipo que evalué la aplicación contra ataques de inyección. La prueba solo fue realizada en forma local encontrando zonas vulnerables en estas aplicaciones. Por lo tanto, recomiendan implementar controles como filtros y validaciones de ingreso de datos y cumplir con la arquitectura REST para desarrollo de aplicaciones Web.

(Hernández Reveles, 2011), presento la tesis de **"Identificación y Clasificación de las Mejores Prácticas para evitar la Inyección SQL en Aplicaciones Desarrolladas en PHP y PostgreSQL";** establece las vulnerabilidades frecuentes que presentan las aplicaciones web por errores de desarrollo. Al final muestra las técnicas manuales y herramientas básicas para el ataque de Inyección según el "Estándar de Verificación de Seguridad en Aplicaciones" propuesto por OWASP.

2.1.2 Antecedentes Nacionales

(Valverde Chavez, 2017) presento la tesis de **"Los riesgos de seguridad de Websites y sus efectos en la gestión de información de medianas empresas de Lima Metropolitana ",** identifica las vulnerabilidades que presenta los Websites de

las medianas empresas, considera las buenas prácticas propuestas por OWASP donde evalúa los grados de riesgos de seguridad que pueda originar en la administración de la información. Propone técnicas preventivas y correctivas que aminoren las vulnerabilidades de las soluciones Web.

2.2 Bases Teóricas

2.2.1 Aplicaciones Web

Desde décadas atrás las aplicaciones en escritorio han sido muy importantes para el mundo de la informática; aunque su acceso limitado al ordenador, instalación y actualización personalizada y otras desventajas su demanda va aminorando. Aun se siguen utilizando y siguen siendo una alternativa para muchas organizaciones.

Lo contrario sucede con las aplicaciones web han pasado a ser indispensables y/o esenciales para toda empresa; adaptándose a sus necesidades, automatizando sus procesos, simplificando sus tareas y comunicación tiempo real, dinámica, ágil entre clientes, colaboradores y usuarios externos.

Al interactuar con una aplicación web se da de la siguiente manera: Se accede desde cualquier lugar a un portal web alojado a un servidor web mediante un navegador haciendo uso de un equipo (laptop, tablet, móvil, etc.) sin importar el sistema operativo que este maneje. Y el proceso de información es como muestra la figura 7.

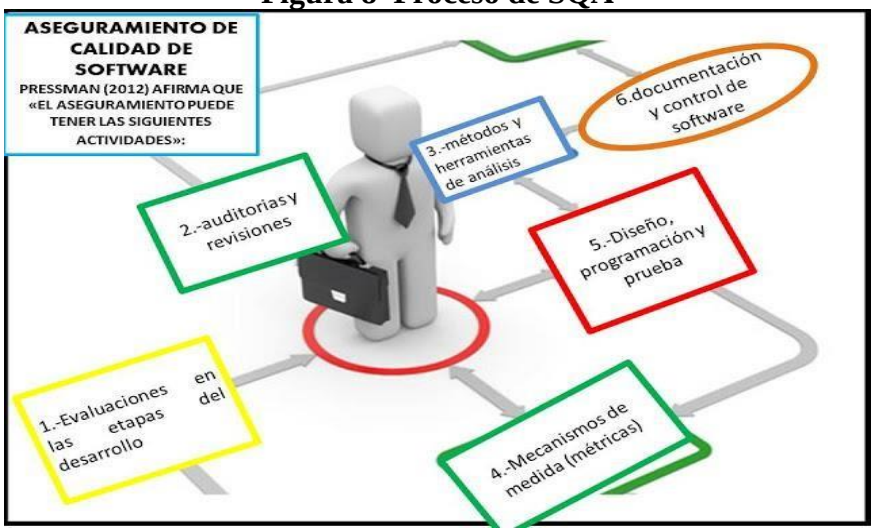
Figura 7 Proceso en Aplicación Web

2.2.2 Aseguramiento de Calidad de Software (SQA)

Es necesario planificar, evaluar, auditar e informar los procesos a realizar durante el desarrollo de software; que estos estén basados en estándares, modelos y patrones que aseguren la calidad de software.

El objetivo de SQA es optimizar los métodos de construcción de software y ser evaluados en cada etapa (figura 8). La responsabilidad de SQA involucra a todo el equipo de desarrollo de software.

Figura 8 Proceso de SQA



Recuperado de http://sistemasunoblog.blogspot.com/2015/07/42-aseguramiento-de-calidad-de-software_9.html

2.2.3 Estándares Internacionales de Desarrollo de Software de Calidad

El equipo encargado de desarrollo de software está en la obligación de realizar los proyectos basándose en los estándares desarrollo de software de calidad durante las fases de análisis, diseño, programación, prueba e implementación, que asegure la calidad del software.

La finalidad es lograr que la aplicación cumpla con las características principales de un software de calidad: Funcionalidad, fiabilidad, usabilidad, eficiencia, eficacia, mantenibilidad, portabilidad y seguridad. Mencionamos algunas características de algunos estándares:

- **ISO 9000:** Es una norma que establece lineamientos de gestión de calidad que las organizaciones deben cumplir para lograr producto o servicios de calidad.
- **ISO 9001:** Los lineamientos están dirigidos al suministro, construcción y mantenimiento del software.
- **ISO 20000:** Los lineamientos se enfocan en como las organizaciones deben de administrar los servicios de TI.
- **ISO 27000:** Los lineamientos establecen como administrar la seguridad de información de una organización.

Tabla 2 Resumen de estándares y modelos relativos a outsourcing de software

Nombre	*Enfoque*	*Objetivo*	*Dimensión de la adquisición y carencias*
Estándar ISO/ IEC 12207	*Outsourcing de software*	Adquirir un producto o servicio que satisfaga las necesidades del cliente.	Identificación de la adquisición de *software* como un proceso principal de las tecnologías de información y definición de las tareas a seguir para llevar cabo una adquisición. *Carece de una descripción y definición completa las mejores prácticas para la adquisición y mejor especificación del enfoque al que se orienta.*
Estándar IEEE 1062	*Outsourcing de software*	Estandarizar el proceso de adquisición de *software* de las organizaciones, teniendo en cuenta aspectos tales como la calidad y la evaluación de la capacidad del proveedor.	Es un marco de referencia del proceso de adquisición de *software*. Su estructura define nueve pasos clave para el proceso de adquisición. *Carece de definiciones y descripciones de las mejores prácticas de ingeniería de software que soporta el proceso de adquisición.*
Modelo CMMI – ACQ	*Outsourcing de software*	Establecer unas prácticas centradas en las actividades necesarias para la contratación del proveedor, el inicio del proyecto de adquisición y la concesión del contrato, así como la gestión de la adquisición de *software*, el sistema de medidas, los criterios de aceptación de entregables del proveedor, entre otros.	Tiene descrita todas las prácticas de las áreas de procesos con una orientación y alineación con las actividades del adquirente. *Carece de guías, herramientas y metodologías que permitan a las organizaciones que adquieren, saber cómo llevar a cabo las prácticas de las áreas de proceso que lo comprenden.*
Estándar eSCM-CL	*Outsourcing de software*	Propuesta de buenas prácticas enfocadas al cliente.	Describe recomendaciones prácticas para el cliente y la gestión de los procesos en relación al cliente. Carece de guía, herramientas, metodologías que apoyen la propuesta. *Carece de métodos claros de gestión de riesgos que tengan en cuenta el proceso de outsourcing para el software*

Recuperado de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-33242010000200010#t2

2.2.4 Buenas Prácticas de Programación

Para desarrollar una Solución Web tenemos muchas opciones de lenguajes de programación a elegir. Cada una de ellas ofrece una variedad de ventajas que aseguran que el programador pueda desarrollar un buen trabajo.

Muchos desarrolladores de sistemas crean soluciones informáticas que funcionen correctamente, que sea ágil, dinámico, de fácil manejo entre otras características comunes. Pero existe un número reducido de programadores que están

capacitados en realizar web no solo con esas características, sino que más importante para ellos es la seguridad y calidad con que debe de diferenciar una web del resto.

Es necesario entonces que los desarrolladores en Perú conozcan la base de las mejores prácticas en desarrollo de soluciones informáticas. Estas buenas prácticas deben ser respaldadas en estándares de seguridad como ITIL, COBIT, CMMI, PRINCE2®, ISO/IEC 20000, ISO/IEC 27000, entre otros.

2.2.5 Organismo de Seguridad de la Información

Las vulnerabilidades de un portal web se presentan porque los desarrolladores de estos servicios no dan la importancia necesaria a los aspectos de seguridad al construir estas aplicaciones; ya sea por descuido, desconocimiento o imprudencia. Para apoyar a que los encargados de proyectos web desarrollen y generen aplicaciones más robustas, a continuación, mencionamos la comunidad más reconocida nivel mundial que nos sugieren lineamientos de buenas prácticas en desarrollo de software:

OWASP: Su principal objetivo es educar a los diseñadores, desarrolladores, arquitectos de software y a las organizaciones en general acerca de las consecuencias de estas vulnerabilidades, así como de la manera en que se pueden prevenir al desarrollar las aplicaciones.

OWASP no es la única organización que hace referencia a estos ataques y los clasifica como peligrosos. Tanto en el WASC (Consorcio de seguridad en

aplicaciones Web) como en el CVE International (Vulnerabilidades y exposiciones comunes), también hacen mención sobre estos ataques y otros.

2.2.6 Modelos Internacionales de mejora de la Calidad de Software

Al cumplir con uno de estos modelos permite asegurar que el software a desarrollar es de calidad. Algunos de estos modelos son:

- **Modelos Deming, Malcolm Baldrige y EFQM:** Son modelos de gestión de calidad basados al ciclo de mejora continua y autoevaluación. Donde va evaluando permanentemente la calidad y la excelencia organizacional. Pueden ser implementados a todo tipo de negocio, donde los resultados van a dar sistemas de aseguramiento de calidad.

- **Modelo CMMI**: Mejora la eficiencia y efectividad de las capas de desarrollo de software. Mide la madurez en una escala de 1 al 5 (Inicial, Repetible, Definido, Administrado y Optimizado). Está orientado a empresas grandes.

- **Modelo TMMI:** Modelo de Madurez de Pruebas Integrado, es un modelo especializado en pruebas de software. Contempla 5 niveles de madurez (Inicial, Gestionado, Definido, Medido y Optimizado)

- **Moprosoft:** Modelo de mejora y evaluación a seguir por las organizaciones a desarrollar software; para gestionar las etapas de desarrollo, implementación y seguimiento.

- **Six Sigma:** Modelo enfocado en lograr el aseguramiento de calidad de software. Consta de cinco fases: Definir, medir, analizar, mejorar y controlar. Hace uso de herramientas estadísticas.

- **MPS:** Es un modelo evaluación y mejora de calidad de proceso de software. Contiene 3 componentes: Modelo de Referencia, Método de Evaluación y Modelo de Negocio.

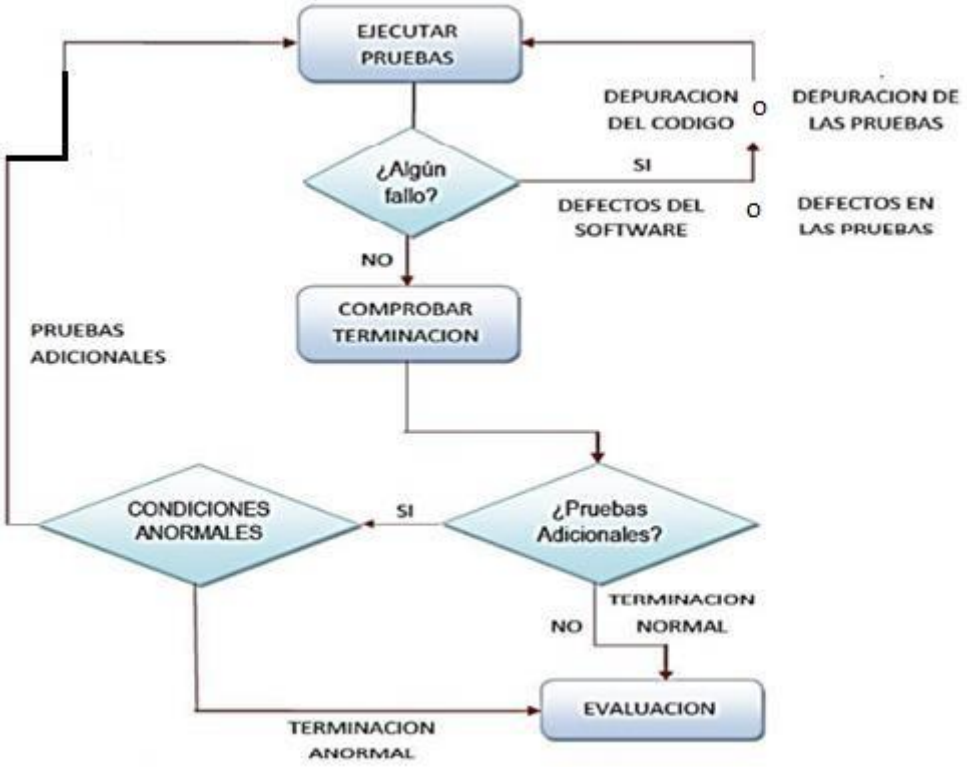
2.2.7 Proceso de Pruebas de Software

En la actualidad las pruebas no se deben realizar una vez finalizada la construcción del software, estos tienen que realizarse en forma paralela a su desarrollo, es decir durante cada etapa de su construcción.

Se debe de planificar y registrar en un historial las incidencias encontradas y como fueron solucionadas; posteriormente que ocurra la misma incidencia saber que se hizo. El análisis de reporte final de cada prueba va a revelar la calidad de software que se está construyendo.

En la figura 9 se muestra el procedimiento a seguir en la ejecución de pruebas de software.

Figura 9 El proceso de Ejecución de pruebas de software



2.3 Definición de Términos Conceptuales

2.3.1 Seguridad de Software

Lograr que el software de ingeniería tenga implementado controles de seguridad y evite ser vulnerable contra todo tipo de amenazas. Para ello durante su desarrollo e implementación fue evaluado y auditado bajo estándares de calidad de software.

2.3.2 Seguridad de Información

Preservación de la confidencialidad, la integridad y la disponibilidad de la información; adicionalmente, otras propiedades pueden también ser involucradas tales como autenticidad, registro, no rechazo y credibilidad. (ISO/IEC 27001, 2005)

2.3.3 Activo

Cualquier cosa que tiene valor para la organización. (ISO/IEC 27001, 2005)

2.3.4 Disponibilidad:

Seguridad de que los usuarios autorizados tienen acceso a la información y a los activos asociados cuando lo requieren. (ISO/IEC 27001, 2005)

2.3.5 Confidencialidad

Seguridad de que la información no está disponible o divulgada a personas o entidades o procesos no autorizados. (ISO/IEC 27001, 2005)

2.3.6 Evento de seguridad de la información

Un caso identificado de un sistema, servicio o estado de la red indicando una posible violación de las políticas de seguridad de la información o falla de control, o una situación desconocida nuevamente que puede ser importante en la seguridad. (ISO/IEC 27001, 2005)

2.3.7 Amenaza

Causa potencial de un incidente no deseado lo que puede resultar dañando al sistema o a la organización. (INDECOPI, 2007)

2.3.8 Vulnerabilidad

Es la debilidad presentada por un activo o grupo de activos que pueden ser explotados por una o más amenazas. (INDECOPI, 2007)

2.3.9 Sistema de Gestión de Seguridad de la Información

Parte del sistema de gestión global, basada en un enfoque de los riesgos de un negocio, para establecer, implementar, operar, monitorear, revisar, mantener y mejorar la seguridad de la información. (ISO/IEC 27001, 2005)

2.3.10 Patrón de Diseño

"Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software." En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (Tedeschi, s.f.)

Existen varios patrones de diseño reconocidos a nivel de todos los lenguajes de programación por la calidad de su estructura al momento de desarrollar software. Algunos de ellos son:

- Prototipado
- MVC (Modelo Vista Controlador)
- Singleton
- Abstract Factory
- Adapter

2.4 Variables

Variable 1: Desarrollo de Sistemas Web

Variable 2: Mejores prácticas contra ataque de inyección

CAPÍTULO III

METODOLOGÍA

Para evaluar el estudio y validar la certeza de la información estadísticas presentadas en el planteamiento de problema. Se realizó trabajo de campo y encuesta tomando como muestra a un grupo de personas relacionados al desarrollo de aplicaciones Web.

La finalidad es demostrar el porcentaje de personas implicadas en el desarrollo de sistemas que no cumplen con los lineamientos de seguridad y no logran obtener una aplicación de calidad.

3.1 Tipo y Diseño

3.1.1 Tipo: Explicativo

El tipo explicativo se enfoca en analizar el problema identificado y saber qué relación hay entre la causa y el efecto. Para el estudio es necesario conocer cómo se formuló la hipótesis para explicar las causas del problema planteado. Existe 2 formas de diseño explicativo: Experimental y observación. En el experimental el investigador va a evaluar su hipótesis. En observación se organiza los datos para poder verificar o refutar la hipótesis.

En la investigación se explica las buenas prácticas que debe conocer todos los integrantes al desarrollo de un proyecto Web; el cual comprende las buenas prácticas de programación, uso de herramientas de evaluación y ataque con Inyección, así aminorar las posibles amenazas contra este tipo de vulnerabilidad. Este estudio trata de demostrar las hipótesis planteadas conforme a las variables identificadas,

cuyos resultados trate de concientizar el rol que debe de asumir el desarrollador para construir un software de calidad.

3.1.2 Diseño:

Para la investigación del tipo explicativo se elige el diseño **experimental** dado que se manipulará las variables identificadas para poder entender los procesos causales. Pero el diseño experimental ofrece diferentes formas de evaluación de hipótesis. Se trabajará con el tipo Cuasiexperimentos, porque se va a manipular la variable dependiente (Mejores prácticas contra ataque de inyección) para poder observar cual es el efecto y/o relación con nuestra variable independiente (Desarrollo de Sistemas Web).

Es decir, que sucede si desarrollas una aplicación Web sin tomar en cuenta las mejores prácticas. La pregunta es: ¿el software es seguro? ¿Es un software que cumple los modelos y estándares de calidad de software? entre otras interrogantes.

3.2 Población y Muestra

3.2.1 Población

La población seleccionada para la investigación está enfocada en las personas involucradas en el desarrollo de proyecto web (jefe de proyecto, analista de sistema, programador, diseñador, administrador base de datos, entre otros), a los que adquieren el servicio de outsourcing para la construcción de un portal Web y por último a facilitadores y estudiantes de aprendizaje de desarrollo web.

3.2.2 Muestra

En esta investigación la selección de los participantes para el análisis de la muestra estuvo clasificado en 4 grupos:

- Profesionales expertos en desarrollo proyecto Web
- Profesionales juniors en desarrollo proyecto Web
- Docentes / Instructores de desarrollo proyecto Web
- Alumnos de desarrollo proyecto Web (egresados y de últimos ciclos).

Se encuesto a 200 personas de las cuales son 30 profesionales expertos, 50 profesionales junior, 40 docentes y 80 alumnos.

3.3 Técnicas e Instrumentos

3.3.1 Técnicas

Encuesta, esta técnica servirá para analizar, medir y evaluar el conocimiento sobre las buenas prácticas en desarrollo de proyecto web que tiene las personas involucradas en la construcción de estas aplicaciones.

Observación, validar si en la actualidad aún se persiste en el error de malas prácticas contra ataque de Inyección por parte de los involucrados en desarrollo de proyectos Web.

3.3.2 Instrumentos

Se desea evaluar conocimiento y demostrar que aún existen aplicaciones vulnerables a los ataques de inyección para ello los instrumentos seleccionados para la investigación son los siguientes:

- **Cuestionario Estructurado:** Los encuestados de cada grupo compartieron sus experiencias a través de entrevistas personales previos y al finalizar respondieron un cuestionario estructurado compuesta por 10 preguntas.

- **Laboratorio de Testing:** Para validar la técnica de observación, solo se realizó con el grupo de estudiantes y profesionales junior.

  En un laboratorio con los requerimientos básicos de hardware y software en un tiempo de 10minutos tenían que realizar ataque de Inyección a portales Web mediante código básico y herramienta SQLMap.

3.4 Procedimiento o levantamiento de información

El proceso de recolección de datos y demostración de ataque inyección se efectuó de la siguiente manera según el tipo de levantamiento de información:

3.4.1 Encuesta (Cuestionario Estructurado):

- La elección de las personas no fue al azar, teníamos conocimientos previos de las funciones que desarrollaba, se pidió que ingrese al grupo

que pertenecía (experto, junior, docente o alumno) antes de llenar las encuestas.

- Las encuestas se realizaron con cada grupo en diferentes fechas teniendo en cuenta la disponibilidad de tiempo de los encuestados.
- Antes de contestar las encuestas se le explico con detalle los 10 tipos de errores más frecuentes que comete el programador web.

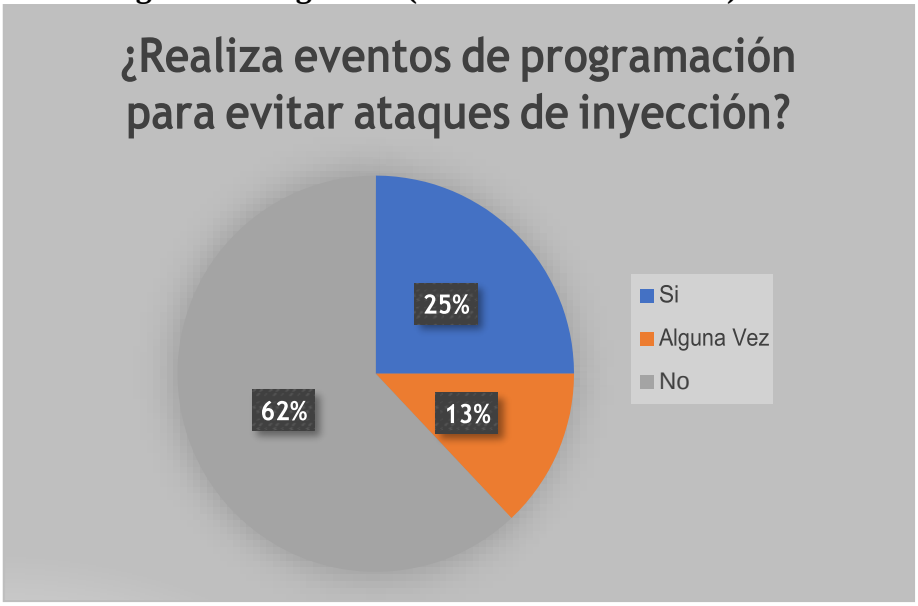
Resultados:

A continuación, mostramos las ilustraciones en modo porcentaje de las preguntas de la encuesta "**Mejores prácticas en Desarrollo de sistemas web contra ataque de Inyección",** que justifican la investigación realizada.

De la pregunta 1: El 62% de los encuestados no implementa controles de seguridad para evitar ataques de Inyección. Este grupo está conformado en su mayoría por estudiantes y profesionales junior.

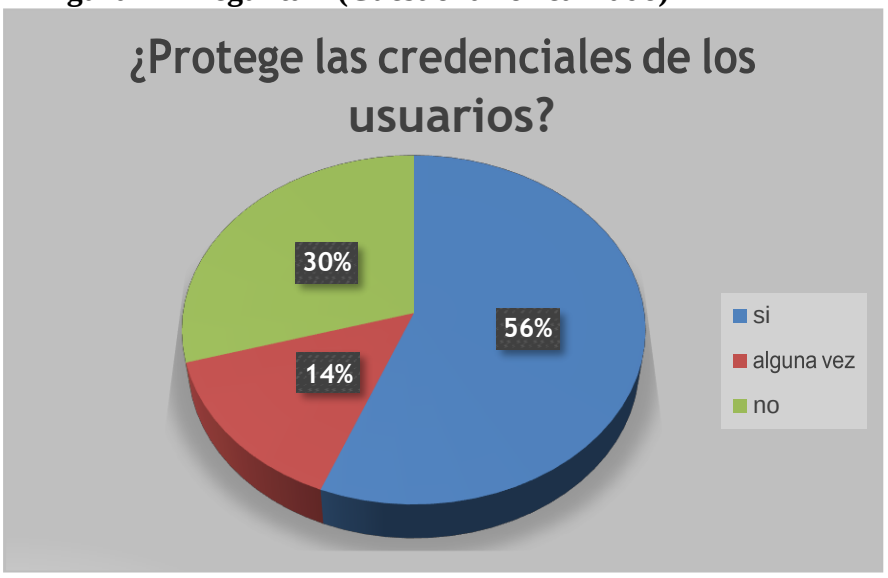
La mayor parte de los profesionales expertos si cumplen con lineamientos de programación segura contra ataque de inyección. Es importante hacer conocer los controles de seguridad a implementar en la programación de aplicaciones web.

Figura 10 Pregunta 1 (Cuestionario realizado)



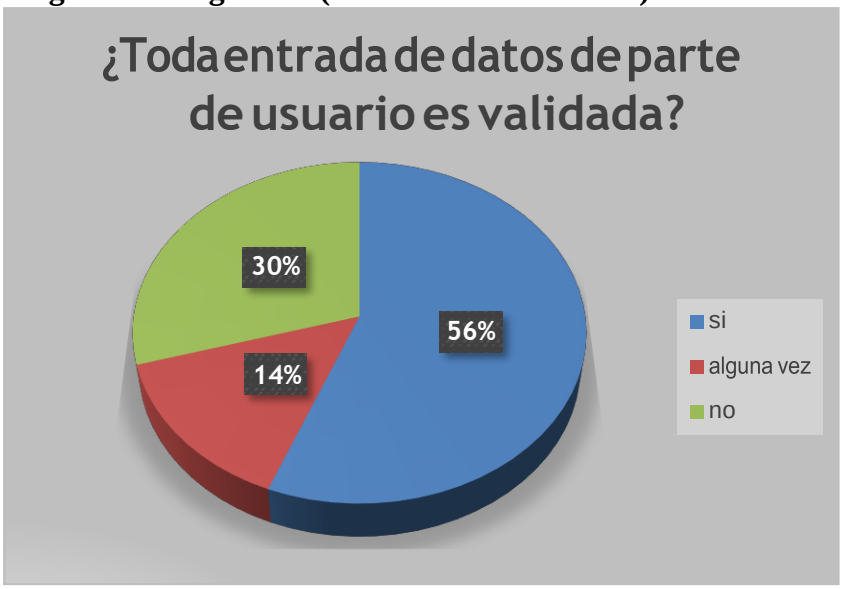
De la pregunta 2: El 65% de los encuestados si usa controles para asegurar las credenciales del usuario que ingresa a una aplicación web. Pero al analizar las respuestas posteriores se deduce que estos controles no siempre son evaluados para validar el nivel de riesgo para el sistema.

Figura 11 Pregunta 2 (Cuestionario realizado)



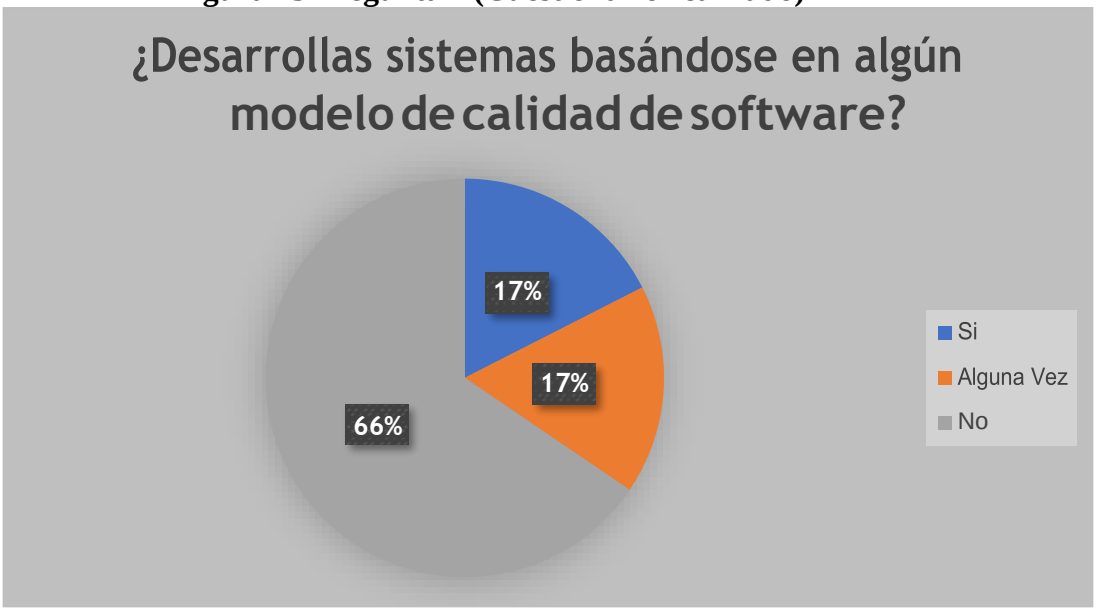
De la pregunta 3: El 69% de los encuestados validan el ingreso de datos, pero al final especificaron que la validación realiza sobre el formato de variable, pero no exactamente sobre códigos de ataque.

Figura 12 Pregunta 3 (Cuestionario realizado)



De la pregunta 4: El 66% de los encuestados al desarrollar software no cumplen con algún modelo de calidad. Por ello que las Organizaciones de seguridad informática cada año informa que se va incrementando los ataques a las aplicaciones Web. Uno de los objetivos del estudio es hacer conocer los modelos de calidad para desarrollar software.

Figura 13 Pregunta 4 (Cuestionario realizado)

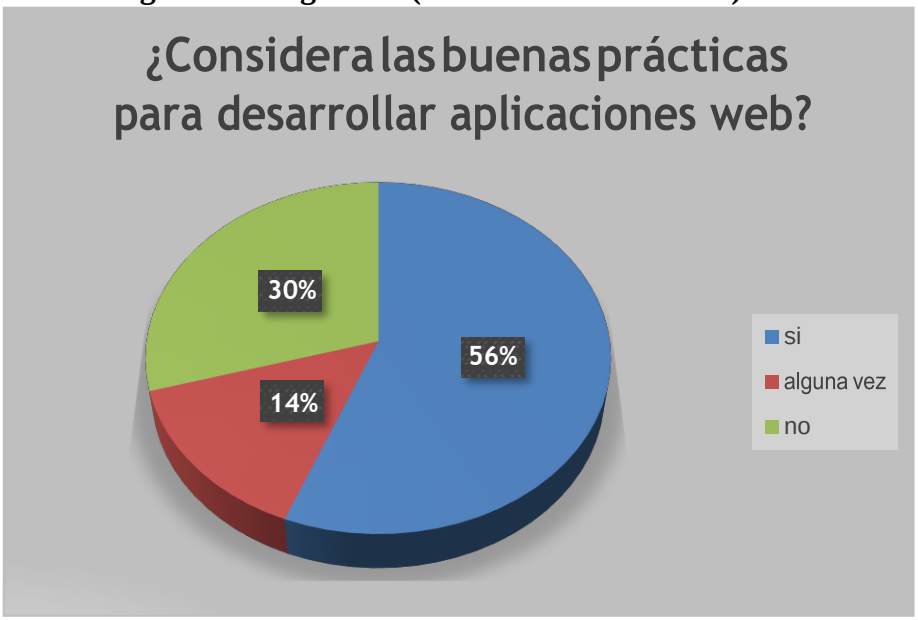


De la pregunta 5: El 47% de los encuestados no almacena el historial de incidencias del sistema este grupo mayoritario se encuentra entre los estudiantes y los profesionales junior. Es importante tener registrado no solo las incidencias también el plan solución con la finalidad que la próxima incidencia saber qué hacer.

Figura 14 Pregunta 5 (Cuestionario realizado)

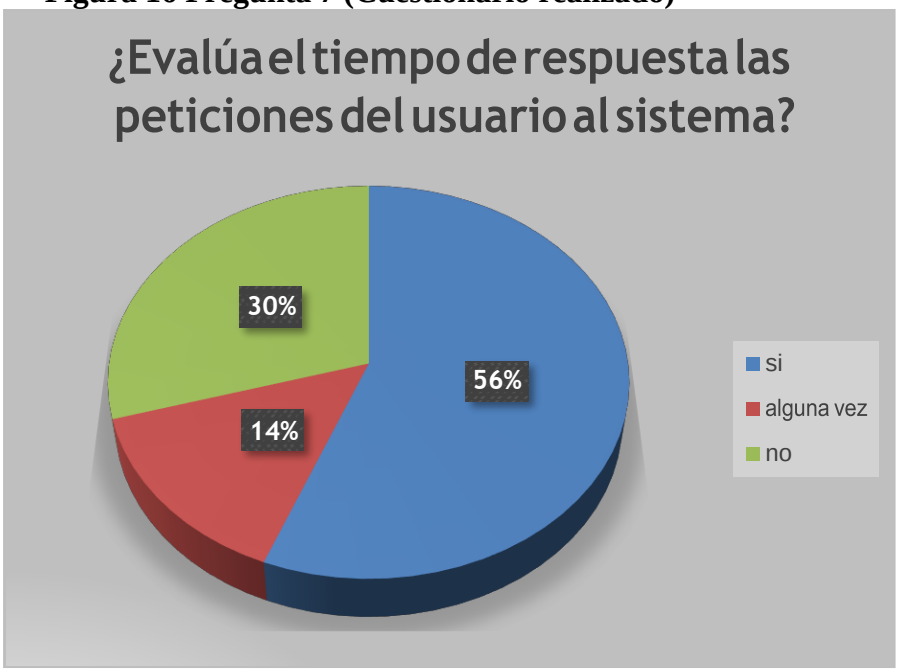
De la pregunta 6: El 37% que consideran las buenas prácticas se encuentra entre profesionales expertos y docentes, el 43% que no consideran se encuentran entre estudiantes y profesionales junior. Es uno de los objetivos del estudio hacer conocer las buenas prácticas durante el desarrollo de aplicaciones web.

Figura 15 Pregunta 6 (Cuestionario realizado)



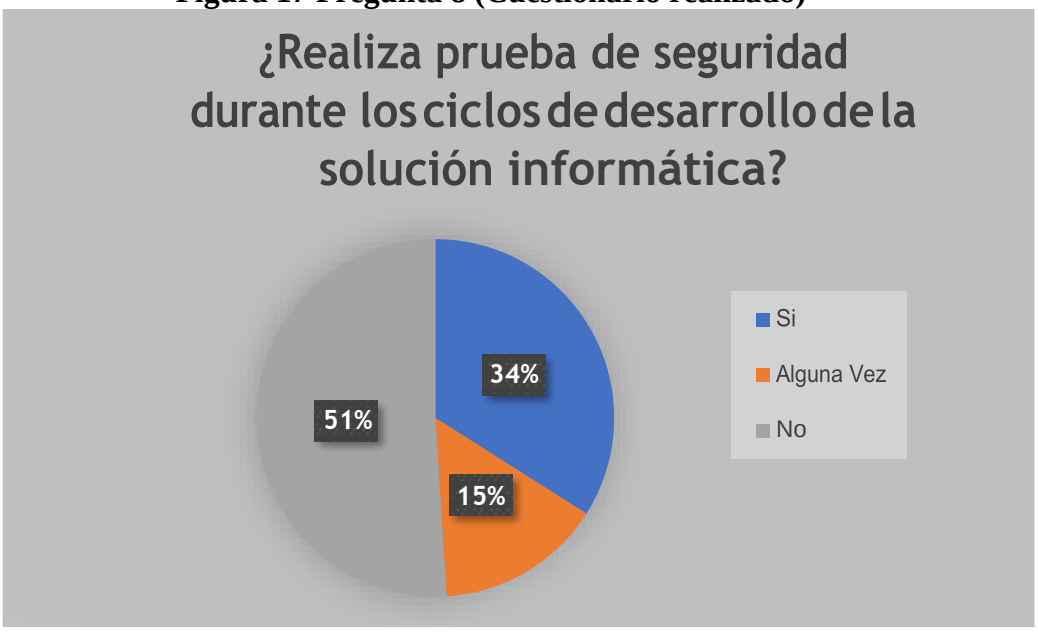
De la pregunta 7: El 56% de los encuestados evalúa el tiempo de respuesta las peticiones del usuario al sistema, afirma el problema que planteamos que se prioriza muchas veces el tiempo de respuesta, no habiendo implementado los controles de seguridad porque que se incrementaría el tiempo de respuesta.

Figura 16 Pregunta 7 (Cuestionario realizado)



De la pregunta 8: El 51% de los encuestados solo realiza pruebas de software antes de ser implementadas mas no en cada fase. El resultado son aplicaciones vulnerables. El estudio propone herramientas no solo para evaluación final del proyecto también para encontrar y explotar vulnerabilidades durante la etapa de desarrollo.

Figura 17 Pregunta 8 (Cuestionario realizado)

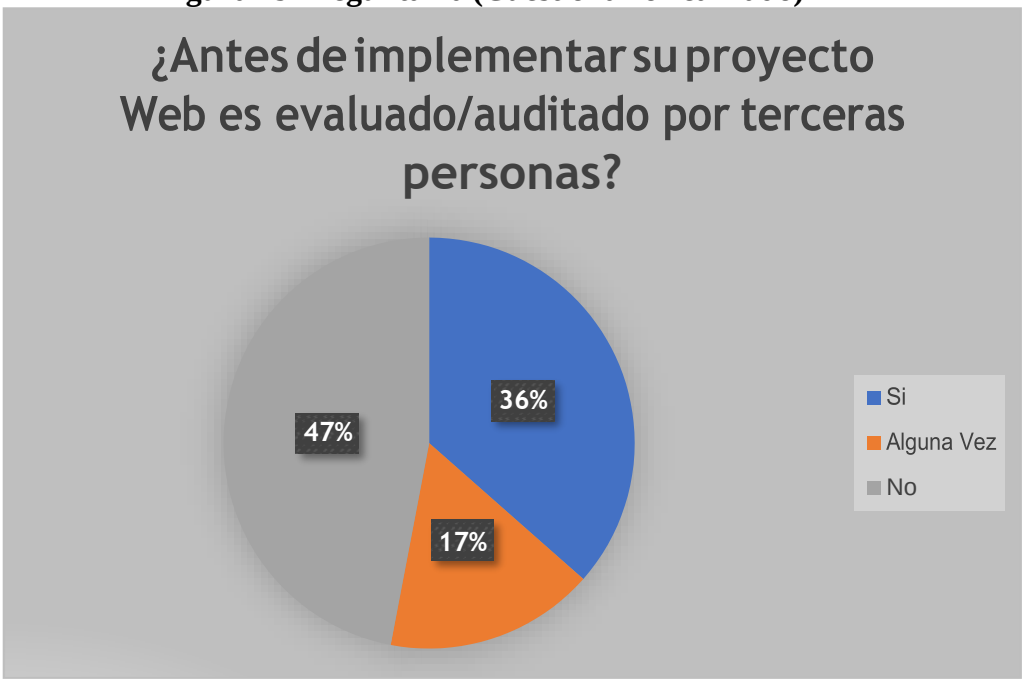


De la pregunta 9: El 66% para desarrollar software no se basa en política de seguridad debido a que no todas las organizaciones tienen implementado. El no tener un SGSI en una organización el trabajo de TI no tiene objetivos y lineamientos de seguridad que cumplir.

Figura 18 Pregunta 9 (Cuestionario realizado)

De la pregunta 10: El 47% que desarrollan proyecto web no audita la aplicación por terceras personas. Antes de implementar una aplicación se recomienda que sea evaluado por un equipo de auditoria que no tenga alguna relación con el equipo desarrollo.

Figura 19 Pregunta 10 (Cuestionario realizado)



3.4.2 Observación (Laboratorio):

- La elección de la muestra fue exclusivamente para los estudiantes y profesionales junior.
- Se explico la técnica básica de búsqueda paginas vulnerables y el ataque de inyección mediante código, se validó que ningún haga uso de una herramienta especializada en estos tipos de ataque.
- Se controlo 10 minutos para que en forma individual presentaran un reporte de los portales que fueron vulnerados.

Resultado:

El resultado promedio final que se realizó en laboratorio durante los 10 minutos cada uno de ellos lograron ingresar en un promedio de 5 páginas haciendo uso de técnica básica de inyección SQL. Podemos afirmar que a pesar de que existen diversos recursos para prevenir estos tipos de ataque aún persiste esa falla en muchos portales web, tal como afirman las estadísticas mostradas de fuentes confiables.

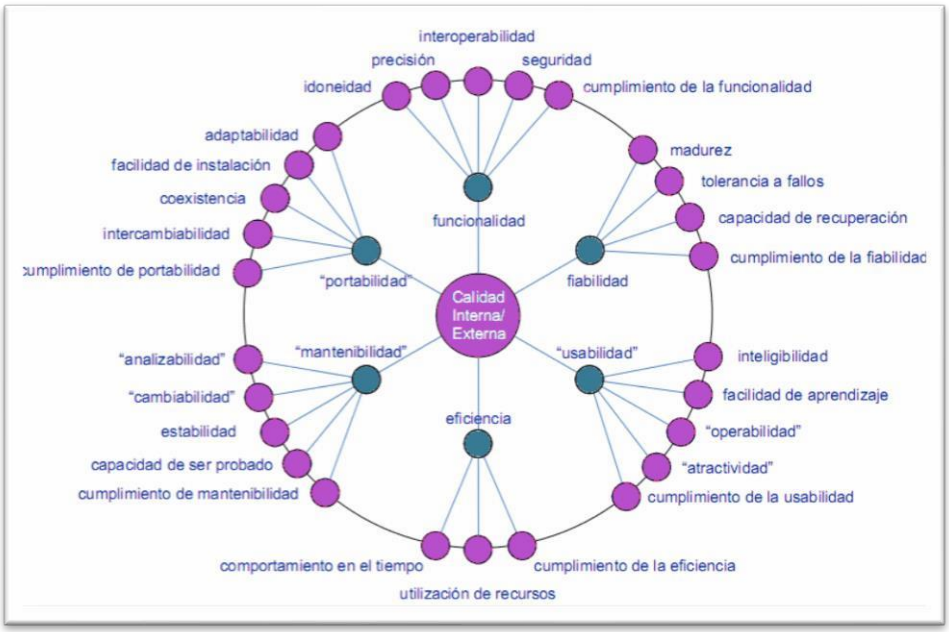
CAPÍTULO IV

RESULTADOS Y ANÁLISIS

4.1 Buenas prácticas para el aseguramiento de la calidad de software

Al decir calidad de software se estará cumpliendo con las características de calidad establecido por la norma ISO 9126, indica que un producto debe cumplir con: Funcionalidad, usabilidad, eficiencia, mantenibilidad, portabilidad y confiabilidad.

Figura 20 ISO 9126



Para asegurar que la aplicación web a desarrollar sea de calidad y que durante su desarrollo se cumpla con las buenas prácticas. El área de TI, las empresas desarrollo de software o cualquier equipo encargado desarrollo de software sus procesos de trabajo y evaluación de procesos siempre deben estar basados en estándar y métricas de aseguramiento de calidad de software y cumplir como está establecido. Si el equipo es de la misma

organización o son terceros (outsourcing) lo que van a desarrollar, debe ellos adaptarse a la política de seguridad establecida por la empresa.

El cumplir con el estándar, métricas y modelo seleccionado va a implicar en el resultado final de la calidad de software a lograr. Los principales beneficios serán:

- El software ha sido desarrollado con las buenas prácticas.
- Aminorar errores de desarrollo de software.
- Validación, verificación y corrección de defectos en software.
- Evaluación de rendimiento de software.
- Que el software trate de cumplir con los pilares de seguridad.
- Trabajo en equipo organizado y delegado según el rol a cumplir de acuerdo al perfil.
- Cumplir con el tiempo de entrega.
- Confianza de los clientes con la empresa.

Las buenas prácticas de seguridad en la construcción y ejecución de aplicaciones web es cumpliendo el SQA. Es decir, el SQA está presente no solo al finalizar el desarrollo de software, sino en todas sus etapas durante el ciclo de desarrollo. Significa que en cada etapa se cumpla los procesos establecidos y luego son evaluadas cada una de ellas. Lo ideal es llegar a un estado óptimo.

Figura 21 Ciclo de Vida Desarrollo de Software

Cada modelo y estándar de desarrollo de software tiene características que aseguran un producto de calidad. La pregunta es ¿Cuál de ellos elegir? esto va a depender de la política de la empresa, de los requerimientos, de los equipos y recursos que pueda tener. En la tabla 3 se muestra la comparación de algunos modelos y estándares de calidad.

B: Modelo de Boehm	D: Modelo de Dromey
MC: Modelo McCall	S: Modelo SATC
C: Modelo C-QM	W: WebQEM
ISO: ISO 9126	F: Modelo FURPS

Tabla 3 Normas y estándares de calidad

Características de calidad	B	D	MC	F	S	C	W	ISO
Facilidad de uso		x	x	x			x	x
Integridad			x		x			
Corrección			x		x			
Confiabilidad	x	x	x	x			x	x
Eficiencia	x	x	x				x	x
Facilidad de mantenimiento		x	x		x	x		x
Facilidad de prueba	x		x					
Flexibilidad			x					
Facilidad de reutilización			x		x	x		
Interoperabilidad			x					
Portabilidad	x	x	x					x
Ingeniería Humana	x							
Fácil de entender	x				x			
Fácil de modificar	x							
Funcionalidad		x		x	x	x		
Performance				x				
Facilidad del soporte				x				
Ambigüedad					x			
Estructura					x			
Documentación					x			
Conformidad						x		

Recuperado de https://marishuyglez.wordpress.com/normas-y-estandares-de-calidad/

4.1.1 Ataques de Inyección

Estos tipos de ataque se origina cuando un usuario envía datos que no son confiables para una aplicación, de modo consulta o comando. Estos datos logran engañar a los controles de seguridad mal configurados o no implementados, y logra acceder a datos que no fueron habilitados para el perfil de usuario que corresponda.

Según los controles de los estándares **ISO/IEC 17799** para evitar ataques de inyección se debe de cumplir el lineamiento de **"Adquisición, desarrollo y mantenimiento de sistemas";** donde especifica lo que es **"Validación de los datos**

de entrada y salida". El no cumplir corre el riesgo de exponer información confidencial almacenado en la base de datos de la aplicación.

Algunas de las causas de Inyección en una aplicación parte desde el diseño y la arquitectura seleccionada, malas prácticas de programación y forma errónea de implementar el patrón desarrollo de software.

Las fallas de inyección son: SQL, NoSQL, OS o LDAP; las características se muestran en la figura 21.

Figura 22 Fallas de Inyección

4.1.2 Tareas previas para realizar pruebas que evidencie problemas de Inyección

Previo a usar las técnicas manuales y/o automatizadas para encontrar problemas de Inyección en aplicaciones web; se recomienda hacer lo siguiente:

- Identificar las páginas que contiene el portal web que permita ingresar datos por parte del usuario final. Sea por método GET o POST.
- Señalar las funciones de javascript que permiten el ingreso de variables.
- Identificar páginas que permitan que los datos sean almacenados en cookies.
- Identificar las paginas cuyas cabeceras http permita el ingreso o modificación de variables.

Cuando el navegador no permita realizar pruebas para evidenciar problema de inyección se realizará:

- Instalación de plug-ins que permita el ingreso de datos que están en método post. Tales como: Form Fuzzer, Tamper Data, Hackbar, SQL Inject Me , entre otros.
- Activación de proxy local, tales como: Paros Proxy, Zed Attack Proxy, Burp Suite, entre otros.

El gestor de base datos SQL interpreta a la comilla simple (') como un delimitador entre el código a consultar y las tablas a manipular. Podemos deducir que algunas aplicaciones web son vulnerables a inyección SQL permitiendo el ingreso de comilla simple en la dirección url o los campos de un formulario.

Los hackers agregan la comilla simple a sus consultas para crear nuevas consultas que vulneren la seguridad de una base de datos. Según el tipo de error que muestre la aplicación ellos deducen que no han validado el uso datos inesperados.

Existen otros operadores que similar a la comilla simple se añaden a las consultas para realizar ataques de inyección, cumpliendo una función especial cada una de ellas. Estos operadores son:

- espacio en blanco
- coma (,)
- punto(.)
- comillas dobles (“)
- /,||
- entre otros

Para realizar la prueba básica que evidencie el error de inyección de un portal web, se puede realizar usando la técnica más común de inyección y se explica porque del error:

Los portales web para autenticar a un usuario solicitan el ingreso de 2 variables que pueden ser: correo-clave, código-clave u otras variables similares.
La aplicación recibe el ingreso de datos de parte del usuario, y envía la consulta al servidor de base de datos y esta es:

select idcodigo, password from usuarios
where idcodigo=10012 and password=cl@ve

Para acceder al sistema, las 2 variables deben ser verdaderos, es decir:

where idcodigo=10012 **and** password=cl@ve
V **and** V (acceso)

Si no tenemos ninguna de las variables, podemos ingresar:

' or'1'='1 **and** ' or'1'='1
(F or V) **and** (F or V)
V **and** V (acceso)

Las técnicas también pueden ser combinadas con:

- **Operador de la Unión:** Cuando se evidencia un error de inyección SQL, usando el operador para combinar otra consulta que visualice información de la base de datos.
- **Booleano:** La finalidad es verificar si la condición es verdad o falso.
- **Basado en el error:** Generar error en la base de datos, para mejorar el ataque de inyección.

En la siguiente tabla se muestra algunas técnicas básicas manuales que puede realizar para evidenciar error de inyección:

Tabla 4 Técnicas básicas inyeccion

Solicitud de aplicación Web	Modo de prueba
http://www.carrito.com/producto.php?id=14	Combinar a la consulta los operadores AND y OR para verificar estado de vulnerabilidad. ?id=10 **AND** 1 = 2
%'or 1=1 union select null, version()#	Visualizar la versión de base de datos
%'or 1=1 union select null,database() #	Para visualizar el nombre de base datos
%'or 1=1 union select null, user()#	Nombre de usuario autenticado

4.2 Identificar las zonas vulnerables de aplicaciones web propenso a los ataques de inyección

4.2.1 Herramientas que evidencie problemas de Inyección

Para evaluar aplicaciones web de gran dimensión en manejo de información y cantidad de páginas, se sugiere automatizar el proceso de búsqueda de vulnerabilidades en Inyección mediante usos de herramientas.

Se debe considerar que estas herramientas no logran identificar las vulnerabilidades en su totalidad, por lo que el conocimiento de pruebas manuales es importante. Las herramientas pueden ahondar puntos críticos que un ser humano no podría identificar. Se mencionará algunas herramientas tanto comercial y como gratuito especializadas en hallar vulnerabilidades de inyección:

a) IBM Rational AppScan

Herramienta comercial que evalúa la seguridad de aplicaciones web. Facilita la gestión de automatizar las evaluaciones que visualicen vulnerabilidades como inyección y otros.

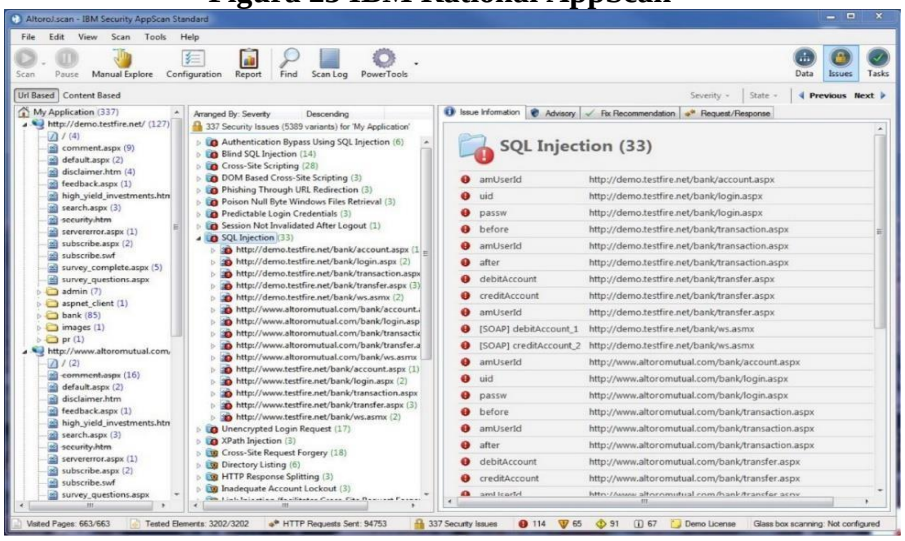
- Emplea métodos de análisis estáticos y dinámicos, dando como resultado un reporte hibrido de la evaluación realizada basados en un promedio máximo de 40 estándares de seguridad, como PCI, OWASP o ISO.
- Muestra alternativas de solución para las vulnerabilidades encontradas.
- url: https://www.ibm.com/products/software

Identificar zonas vulnerables de una o varias aplicaciones con IBM Rational AppScan :

- En vista Explorador, elija 1 o varias aplicaciones
- Seleccione Escanear. Se iniciará el proceso según el tipo de configuración determinado por la aplicación (por defecto) o por el usuario.
- Si desea configurar el modo de escaneo seleccionar Editar configuración.

- Filtrar los resultados que muestre el análisis de las vulnerabilidades de Inyección tal como muestra la figura 23.

Figura 23 IBM Rational AppScan

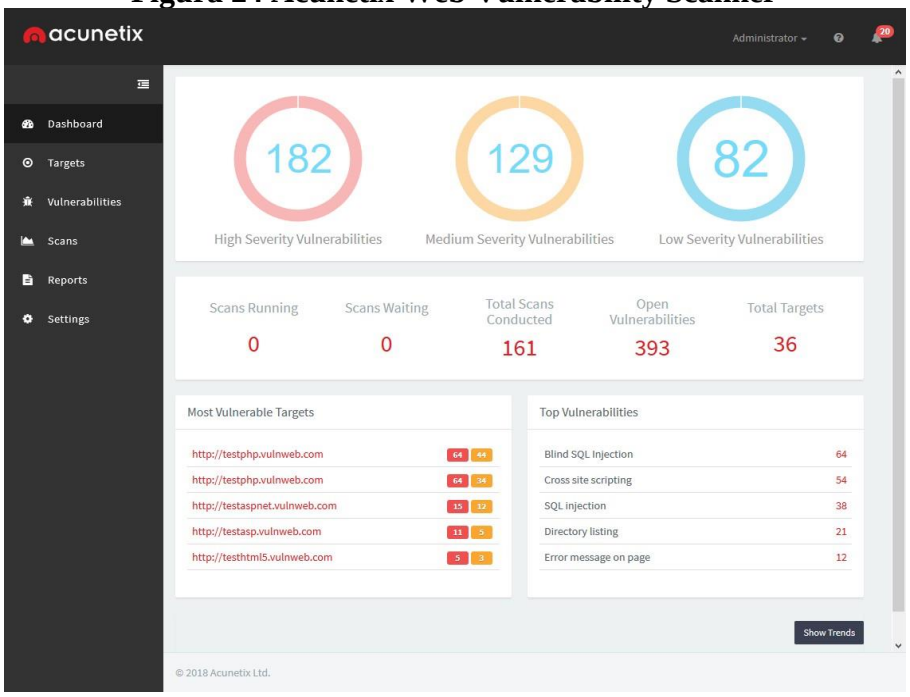


b) Acunetix Web Vulnerability Scanner

Herramienta comercial para realizar pruebas automáticas para inyección y más de 4500 vulnerabilidades web.

- Los reportes cumplen con las nomas de PCI-DSS, ISO/IEC 27001, OWASP, CVSS y otros; ofrece lineamientos de remediación.
- Realiza pruebas de caja gris, prueba de vulnerabilidad fuera de banda y motores de escaneo múltiple.
- Para identificar las vulnerabilidades de Inyección es simple como las otras herramientas. Ingresar la url de página y dar click a escanear. Esperar que muestre un reporte estadístico y descriptivo sobre las vulnerabilidades encontradas y su impacto a la web.
- url: https://www.acunetix.com/

Figura 24 Acunetix Web Vulnerability Scanner



c) SQLiX

Es un script desarrollado en Perl, herramienta gratuita para indagar y evidenciar inyecciones SQL. Maneja técnicas como: Inyección de errores condicional, Blind SQL Inyección y mensaje de errores descriptivo.

- Identifica errores de servidores de base de datos como: MS-SQL, MySQL, Oracle y PostgreSQL.
- Tiene un módulo de explotación de prueba para demostrar como terceros podrían realizar inyección y acceder a datos confidencial.
- Sintaxis de comando de inyección para MySQL, PostgreSQL

  $ perl SQLiX.pl -file crawling -all -v=2 -exploit

  Ejemplo:

  $ perl SQLiX.pl -crawl="http://www.paginavulnerable.com/" -all -v=2 -exploit

- url: https://www.owasp.org/index.php/Category:OWASP_SQLiX_Project#tab=Main

Figura 25 SQLiX

d) Paros Proxy

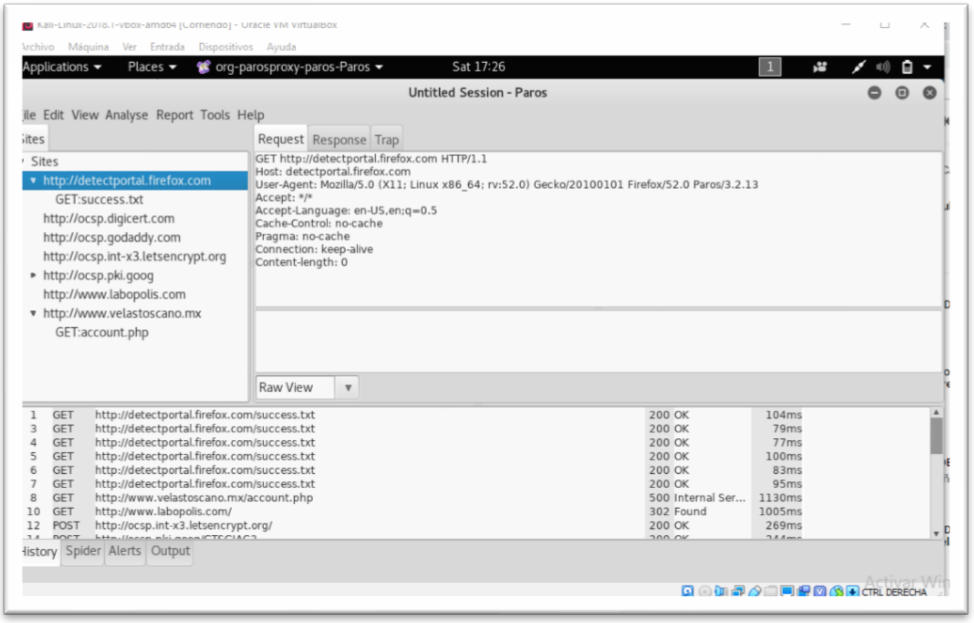
Un proxy HTTP/HTTPS gratuito desarrollado en Java que evalúa vulnerabilidades de Inyección y XSS que puede manipular cualquier tráfico sobre la marcha. Se va a automatizar técnicas de evaluación como: Búsqueda, escáner y filtro

- Plataforma: Windows y Linux
- Escaneo inteligente
- Edición y visualización de mensajes HTTP en ejecución
- Esta herramienta viene incluida también en Kali Linux.
- url: http://www.parosproxy.org/

Identificar zonas vulnerables de una o varias aplicaciones con Paros Proxy :

- Configurar el navegador web para realizar una conexión proxy con el servidor Paros. Modo host local(127.0.0.1) y puerto TCP 8080.
- Validar la sincronización del navegador con el servidor Paros.
- Paros muestra las opciones: historial, spider, alertas de vulnerabilidad notadas. Los resultados dependen de la opción seleccionada.
- Da opción para automatizar el escaneo de pruebas al sitio de web. Se puede personalizar esta opción.

Figura 26 Paros Proxy



4.2.2 Análisis Estático de código vulnerable a Inyección SQL

Son procesos de evaluación que se realiza sin necesidad de ejecutar la aplicación (SAST), es el análisis del código fuente que sucedería al ser ejecutado. El análisis se puede realizar en forma manual en aplicaciones medianas, pero para grandes aplicaciones es necesario el uso de herramientas o scripts. La finalidad es la evaluación de los códigos que van a implicar en la seguridad de la aplicación.

Consultas como estas son vulnerables porque permite el ingreso directo de comandos SQL que afecte el contenido de base de datos:

- PHP

  $query = mysql_query(“select * from tabla from where columna= ’$_Get[parametro]’ ”);

- . Net

  SqlCommand cmd= new SqlCommand("Select * from tabla from where columna= ’parametro’ “,cn);

- Java

  ResultSet rs = st.executeQuery("select * from tabla where columna="+ parametro);

Si a las consultas dinámicas presentadas se les invocaría desde la aplicación mediante procedimiento almacenados también serían vulnerables si han sido programados en forma errónea como el siguiente procedimiento:

```
create procedure ingresar
@param text
```

As

Select * from tabla from where columna=@param

El desarrollador tiene que evaluar cada función que permite el ingreso de parámetros que este interactuando con la base de datos porque pueda causar riesgos de alto nivel. Se debe entonces realizar pruebas unitarias, de sistema y de aceptación para evaluar el ingreso de datos. Esta evaluación va a implicar mucho tiempo y realizar tareas minuciosas; también se podría automatizar mediante herramientas o scripts.

a) AppCodeScan

Herramienta gratuita para realizar prueba de testing de caja blanca, evalúa el código fuente para detectar vulnerabilidad como inyección SQL, XSS y otros. Su función es rastrear el código hasta encontrar las líneas de código vulnerable.

Trabaja de la siguiente manera:

- Analiza la carpeta que contenga las reglas de programación. A base de la información realiza el rastreo de vulnerabilidades en cada línea de código
- Code Tracer realiza el seguimiento de ruta de la variable, método o función que fue ejecuta para ser analizada.
- Requisito: Instala .NET Framework para su ejecución.
- Plataforma: Window
- url: https://www.blueinfy.com

b) LAPSE: Web Application Security Scanner para Java

Herramienta gratuita para auditar la seguridad con respecto a la codificación de aplicaciones Web desarrollado en Java. Muestra la vista de la fuente, vista del fregadero y vista del rastreador de procedencia de las vulnerabilidades como SQL Inyección, manipulación de parámetros y cabeceras, envenenamiento de cookies y otros.

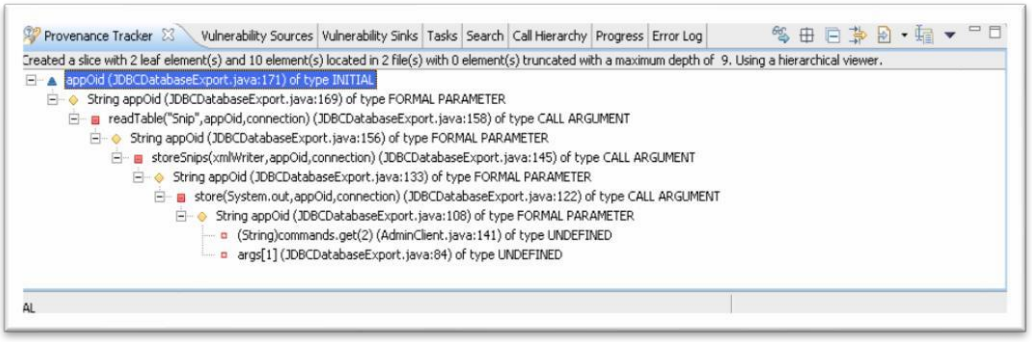
- Plataforma: Window y Linux
- url: https://suif.stanford.edu/~livshits/work/lapse/download.html

Identificar procedencia de la vulnerabilidad con LAPSE :

Si recién está comenzando con LAPSE, se recomiendan los siguientes pasos:

- Abrir las interfaces de LAPSE en Eclipse
- Para ubicar las fuentes de la vulnerabilidad se elige el icono de error) del interfaz fuente.
- A continuación, ubicar los receptores de vulnerabilidad pulsando el icono de error) del interfaz de sumidero.
- Marcar una variable particular para observar de dónde proviene, tal como indica la figura 27 .

Figura 27 Búsqueda de Vulnerabilidad con herramienta LAPSE



c) Herramientas para realizar Análisis Estático

A continuación, la tabla 5 muestra otras herramientas para realizar análisis estático en portales web:

Tabla 5 Herramientas para auditar códigos

Herramienta	Tipo	Plataforma	Lenguaje Programación
Ounce Labs	Comercial	Windows, Solaris y Linux	Visual Studio y Java
CAT.NET	Gratuito	Windows	C #, Visual Basic .NET, J #
CodeSecure	Comercial	Windows	PHP, ASP y .NET

4.3 Técnicas y Herramientas de explotación en Inyección de SQL

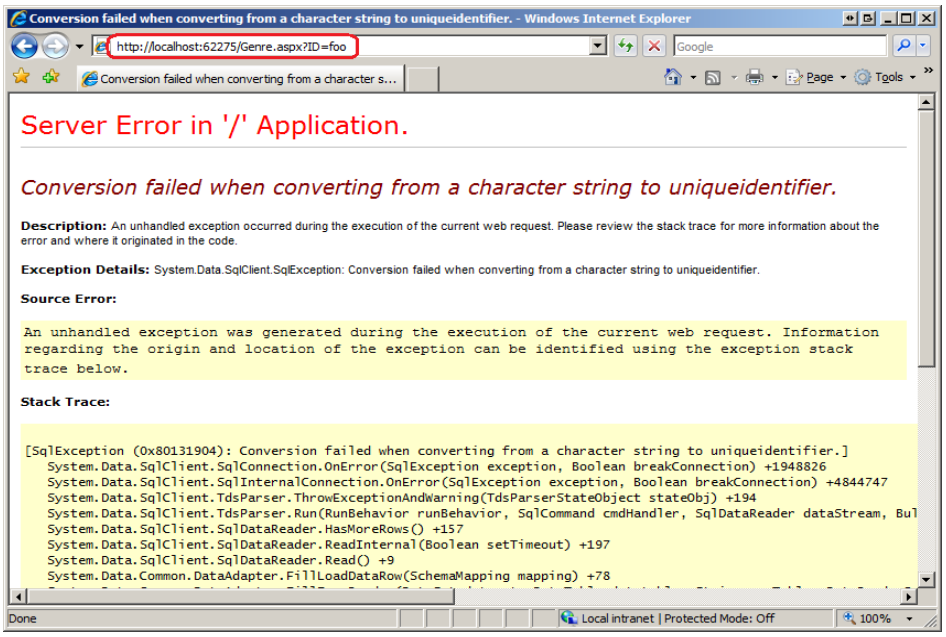
Al encontrar un punto inseguro para realizar inyección SQL en una aplicación Web, el hacker intentara ser uso de sentencias select, insert, update y delete para vulnerar la base de datos.

Para visualizar la tecnología de DBMS simplemente es ver el mensaje de error que muchas veces es detallado tan solo ingresados valores no permitidos. Información como muestra la imagen va a ser de vital importancia para el intruso.

Figura 28 Portal 1 con error

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark before the
character string ''.
/target/target.asp, line 113
```

Figura 29 Portal 2 .net con error



4.3.1 Técnicas de explotación Inyección SQL con declaraciones UNION

Es un comando muy usado en ataque de inyección, finalidad es unir uno o más consultas a la variable ingresada que solicita la web. Para lograr que funcione este ataque se considera:

- Las consultas anidadas tendrán la misma cantidad de columnas.

- Las columnas por mostrar de cada consulta deben ser del mismo tipo de variable.

La técnica más fácil de lograr saber cuántas columnas arrojara la consulta principal es usando el ORDER BY incrementando el número de columnas.

?id=15+order+by+4

?id=15+order+by+5

?id=15+order+by+6

Al saber el número de columnas y los tipos de variables a mostrar. El atacante empieza a realizar diversos tipos de consultas de acuerdo a la sintaxis del gestor de base de datos. A continuación, en la tabla 6, se muestra algunos comandos que añadir a la consulta y lograr obtener información de la base de datos a través de los controles y cabeceras de ingreso de la aplicación Web:

Tabla 6 Técnicas de inyección con Unión

Técnica	Objetivo
union select 1,version() #	versión
union select 1,user() #	usuario
union select 1,database() #	nombre base de datos
union select 1,current_user() #	nombre de usuario y el del host
union select 1,connection_id() #	ID de una conexión
union select 1 -- **union select 1,2--** **union select 1,2,3--**	en forma sucesiva se añade el numeral hasta encontrar la columna vulnerable

union select 1 from all_tables	tablas en Oracle
union select 1 from information_schema.tables	nombre tabla Mysql
union select 1, session_user() #	usuario de sesión

4.3.2 Herramientas para automatizar explotación en Inyección de SQL

Para realizar estos ataques en forma manual se requiere ingresar muchas solicitudes para obtener información de forma no permitida de una base de datos. Para estos casos también existe herramientas que se pueden automatizar y obtener la información esperadas de forma manual.

a) SQLMap:

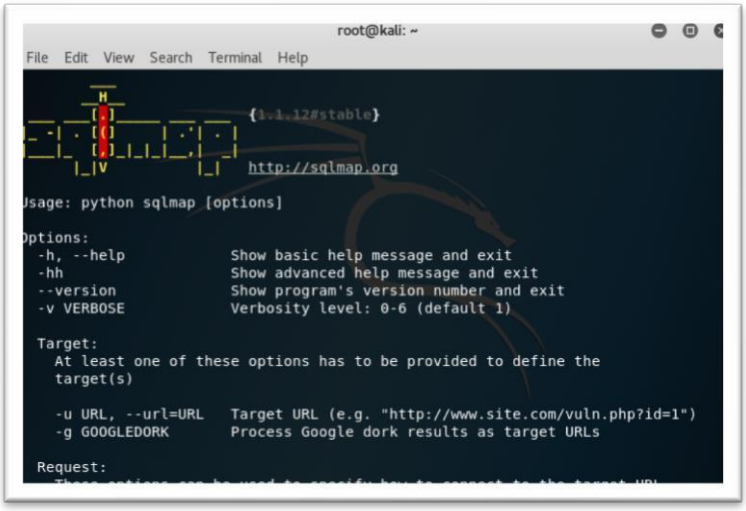
Herramienta gratuita para automatizar la detección y explotación de vulnerabilidades de inyección SQL y toma de control de servidores de base de datos. Desarrollado en phyton. Sus características principales son:

- Ataque a DBMS: MySQL, Oracle, Postgree SQL, SQl Server, SQLite y entre otros.
- Técnicas de ataque de Inyección SQL: Basado en errores, consultas union, consultas anidadas, fuera de banda y ataque ciego.
- Muestra como resultados listado de usuarios, contraseñas encriptadas o texto plano, base de datos, tablas, columnas y roles.
- Descifrar contraseñas encriptadas mediante uso de diccionarios.
- url: http://sqlmap.org/

Tabla 7 Comandos SQLMap

Comandos	Descripción
sqlmap --dbms=mysql -u "direccion de url" --dbs	Listar base de datos
sqlmap --dbms=mysql -u "direccion url" -D "base datos" --tables	Listar tablas
sqlmap --dbms=mysql -u "direccion url" -D "base datos" -T "tabla" --dump	Dumpear la información de tabla
sqlmap --dbms=mysql -u "direccion url" --os-shell	Shell de sistema
sqlmap --dbms=mysql -u "direccion url" --sql-shell	Shell SQL
sqlmap --dbms=mysql -u "direccion url" -D "base datos" --sql-query "consulta"	Ejecutar consulta base datos

Figura 30 SQLMap



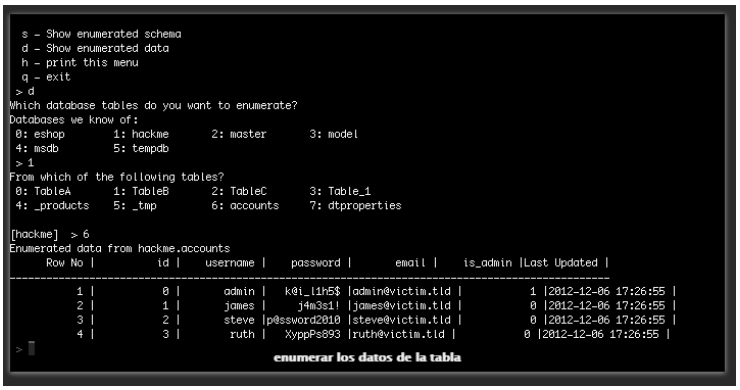
b) <u>SQLNinja:</u>

Herramienta gratuita desarrollada en Perl, que puede ser ejecutado en plataforma UNIX, pero no en Windows. Automatiza la explotación de

inyección SQL a servidores de base de datos de SQL Server. Sus características son:

- Huella digital del servidor SQL
- Ejecuta ataque de fuerza bruta para romper contraseñas
- Carga de archivos ejecutable haciendo uso de las solicitudes de HTTP
- Extracción de Datos
- Maneja técnicas que confunden monitoreo de red de herramientas como IDS / IPS / WAF.
- url: http://sqlninja.sourceforge.net/

Figura 31 SQLNinja obteniendo contenido de tabla



c) The Mole:

Herramienta gratuita desarrollado con phyton especializada en la explotación automatizada en inyección SQL, mediante técnicas de unión o booleanas. Las características son:

- Ataque a DBMS: MySQL, Postgres, SQL Server y Oracle
- Explotar inyecciones a través del método GET, POST y Cookies.

- Compuesto por un shell con comandos propios para manipular la base de datos de la aplicación.
- Soporte para filtros de consulta IDS/IPS.
- url: http://themole.sourceforge.net/

4.3.3 Explotación de Inyección mediante SQL Ciega

También conocido como Blind SQL Inyección. Estos ataques se realizan a las páginas que no retornan error cuando el usuario envía en forma incorrecta los parámetros solicitados y solo responde cuando la consulta es correcta. Estos ataques están basados en expresiones booleanas. Usan técnicas como de diccionarios, ataque de fuerza bruta, temporización entre otros.

Basado en Contenido:

Realizar ataque booleano (verdadero o falso)

http://ataque.com?idemp=1 Resp: Robert Williams

http://ataque.com?idemp=2 Resp: Maria Smith

Se sigue cambiando el código, a continuación, podemos deducir que solo existe 10 datos en la tabla

http://ataque.com?idemp=11 Resp: Item no encontrado

Ahora agregaremos una consulta boolean

http://ataque.com?idemp=1 and 1=1 Resp: Robert Williams

http://ataque.com?idemp=1 and 1=0 Resp: Item no encontrado

Cuando añadimos “and 1=1” retorna una respuesta, se deduce que la pagina es vulnerable para agregar otros tipos de consulta según los permisos admitidos para el perfil de usuario registrado.

Basado en Tiempo:

Lograr que la base de datos permanezca pausada por un intervalo de tiempo ingresado y luego responda la consulta ingresada.

La aplicación será vulnerable si tarda 15 segundos.

http://ataque.com?idemp=1 and if(1=1, sleep(15),false)

4.3.3.1 Automatización de explotación Blind SQL Inyección

El ataque Blind SQL Inyección también se puede automatizar mediante herramientas que se basan en técnicas de inferencia o canales alternativos. Algunas de ellas son:

a) BSQL Hacker

Herramienta gratuita puede ser usado por usuarios principiantes o experimentados, donde se podrá automatizar Blind SQL Inyección a DBMS (MS-SQL, Oracle, Ms Access, Postgree y MySQL).

- Tipos de ataque: Inyeccion Blind SQL, Deep Blind, Inyeccion SQL basado en errores.
- Ataque multihilo para agilizar y mejorar la búsqueda.

- Acepta URL protegidos de SSL o de SSL con certificados no válidos.
- Soporte Consola o Grafico
- Requisitos: Windows (.NET Framework)

url: https://github.com/Neohapsis/bbqsql/

b) SQLBrute

Herramienta gratuita desarrollado en phyton. Automatiza el ataque de fuerza bruta a DBMS (MS-SQL y Oracle) mediante Blind SQL Inyección. Adopta a exploits basado en errores y tiempos.

- Maneja subprocesos múltiples
- Métodos: Inferencia búsqueda binaria basada en el tiempo; inferencia respuesta de base modificado búsqueda binaria
- Soporte consola

url: https://github.com/GDSSecurity/SQLBrute

c) SQL Power Injector

Herramienta gratuita creada en .Net 1.1 ayuda a encontrar y explotar Blind SQL Inyección a portales Web. Automatiza la inyección de múltiples subprocesos.

- Sistemas operativos Windows, Unix y Linux
- Ataque a DBMS: MySQL, SQL Server, Oracle, DB2 y Sybase.

- Inyección ciega de SQL: Tiempo de retardo y comparación de resultados
- Multihilo (configurable hasta 50)
- Resultado en tiempo real

url: http://www.sqlpowerinjector.com/download.htm

4.3.4 Explotación ataque Inyección NOSQL

El ataque de inyección SQL es la técnica más utilizada en inyección. Caso contrario son los ataques a base de datos NoSQL, son base de datos no relacionales que almacenan grandes cantidades de información. La base de dato al no usar sintaxis SQL en su creación son potencialmente vulnerables a los ataques de inyección.

Los tipos de base de datos NoSQL más conocidos son MongoDB, Cassandra, Hadoop, CouchDB, Redis entre otros. Estas bases de datos los podemos ver en aplicaciones web como: Web logs, redes sociales, Wikipedia, etc. Los ataques de inyección NoSQL se dan por:

- Usar lenguajes imperativos en lugar de declarativos.
- La inyección se ejecuta donde se analiza y evalúa la consulta.
- Usar formatos estandarizados (XML, JSON, LINQ) vulnerables a código malicioso de caracteres especiales.

Ejemplo:

XML <> &;

JSON /{}:.

Herramientas para su explotación se nombran algunas de ellas:

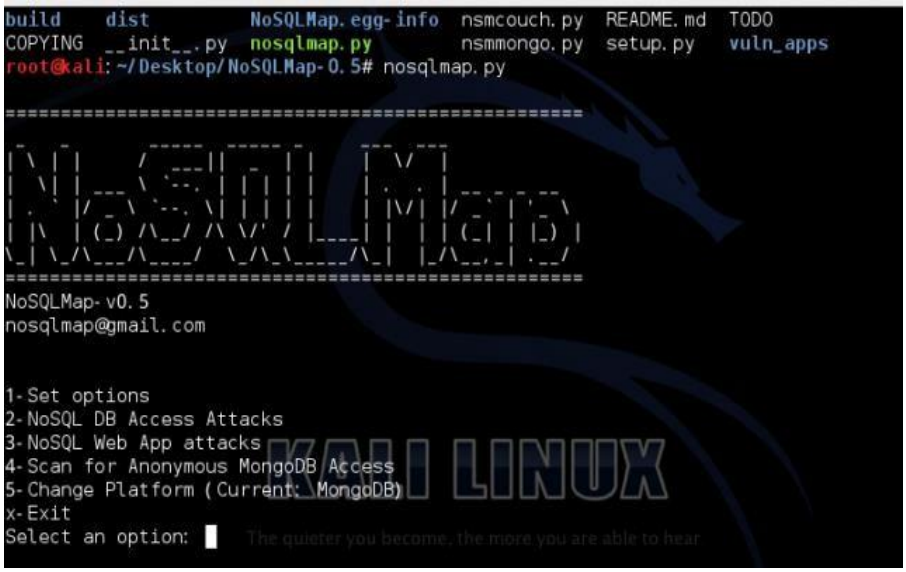
a) NoSQLMap

Herramienta gratuita desarrollada en Phyton de código abierto, se usa para automatizar y auditar los ataques de inyección y vulnerar los puntos críticos por mala configuración de base de datos NoSQL y la aplicación Web.

- Vulnerar base de datos MongoDB
- Almacenamiento y recuperación de datos
- Presenta las opciones: Ataque a base de datos NoSQL, Ataque a aplicaciones web NoSQL y escaneo para el acceso anónimo de MongoDB

url: https://github.com/codingo/NoSQLMap

Figura 32 Opciones de ataque de NoSQLMap



b) Arachni

Herramienta gratuita de fuente libre, para automatizar pruebas de penetración a aplicaciones web como: inyección NoSQL, inyección SQL, inyección código y otros tipos de inyección.

- Multiplataforma: Windows, Linux y Mac OS X
- Escanear uno o varios servidores
- Realizar auditorías mediante scripts
- Entorno de navegador integrado

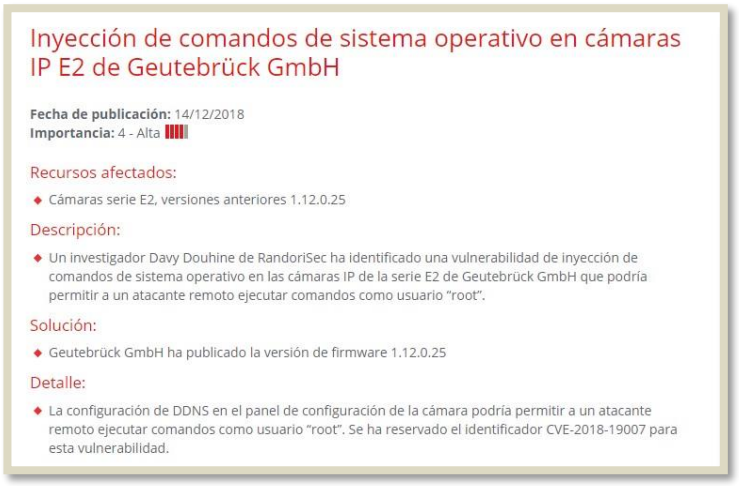
url: http://www.arachni-scanner.com/

4.3.5 Explotación inyección a sistema operativo

A través de una aplicación web también se realiza consultas al servidor de base de datos que da acceso al sistema operativo. Como manipulación (lectura, escritura y almacenar) de archivos y ejecución de comandos; sino se implementa controles de seguridad podría ser un punto crítico para cualquier ataque.

La aplicación en algunos procesos va a actuar como un Shell, y si no se ha limitado la entrada de comandos estos pueden ser mayor riesgo que un ataque inyección SQL porque no solo se tendrá acceso a la base de datos también al sistema operativo. Como se puede ver la figura 28 que ocurrió cuando se accedió al sistema operativo de una red mediante ataque de inyección.

Figura 33 Caso de ataque Inyección OS



Inyección de comandos de sistema operativo en cámaras IP E2 de Geutebrück GmbH

Fecha de publicación: 14/12/2018
Importancia: 4 - Alta

Recursos afectados:

- Cámaras serie E2, versiones anteriores 1.12.0.25

Descripción:

- Un investigador Davy Douhine de RandoriSec ha identificado una vulnerabilidad de inyección de comandos de sistema operativo en las cámaras IP de la serie E2 de Geutebrück GmbH que podría permitir a un atacante remoto ejecutar comandos como usuario "root".

Solución:

- Geutebrück GmbH ha publicado la versión de firmware 1.12.0.25

Detalle:

- La configuración de DDNS en el panel de configuración de la cámara podría permitir a un atacante remoto ejecutar comandos como usuario "root". Se ha reservado el identificador CVE-2018-19007 para esta vulnerabilidad.

Recuperado de https://www.incibe-cert.es/alerta-temprana/avisos-sci/inyeccion-comandos-sistema-operativo-camaras-ip-e2-geutebruck-gmbh

La inyección OS tiene 2 finalidades:

- Ejecutar un programa que tome control del sistema.
- La aplicación permita la ejecución interna de un programa o comando.

Las técnicas de inyección a sistemas operativos son similares a inyección SQL, en lugar de añadir consultas se añade comandos. Pueden usar los rootkits (aplicación que oculta un programa maligno en el sistema) para ingresar en forma ilícita y oculta al sistema operativo. Los puntos más vulnerables aparte de los mencionados en inyección SQL son los controles que permiten cargar o leer archivos. Para remediar estos ataques tener una lista blanca de comandos permitidos y lista negra los que no están permitidos no deben de ejecutarse. Para estos casos la validación de entrada no asegura el ataque de inyección OS.

a) COMMIX

Herramienta de inyección gratuita desarrollado en phyton para la explotación de comandos a sistemas operativos.

- Requerimientos: Python
- Sistemas operativos: Linux, Unix y Windows
- Explotar aplicaciones desarrollados en C, C++, Phyton, PHP y Java.
- Preinstalados en Linux, Parrot Security OS y Backbox
- Descargar en: https://github.com/commixproject/commix

4.3.6 Explotación a Inyección LDAP

LDAP se encarga almacenar información de usuarios, sitios y otros servicios de directorio pertenecientes a un dominio o una red. Mediante una aplicación Web haciendo uso de LDAP el atacante podrá tener acceso a los recursos de la red.

Inyección LDAP es el ataque de lado servidor que permitiría manipular los recursos de dominio. Las técnicas de ataques son similares a Blind SQL Inyección donde se espera el resultado sea verdadero o falso para retornar una respuesta y poder manipular los parámetros a enviar haciendo uso de meta caracteres para filtra la búsqueda LDAP.

En una aplicación que no cumplió con los controles de seguridad contra ataque de inyección LDAP, el ataque puede ser el siguiente:
http://ataque.com/index.php?nombre=**Andres**

Se va a cambiar la variable de ingreso (**Andres**) por:
http://ataque.com/index.php?nombre= **x) (|(cn=***

Esta consulta retornara una lista de los usuarios registrados.

Estos ataques se dan por 2 razones:

- Las variables por enviar a la consulta LDAP no son enviadas en forma parametrizada ni validadas.
- Realizan autenticación LDAP a los usuarios de la aplicación.

Para localizar vulnerabilidad de inyección se recomienda:

- Prueba caja negra: Añadir consultas LDAP no permitidos.
- Búsqueda de código: Ubicar usos de la API de LDAP en el código, verificar que el ingreso de datos es validado.

Figura 34 Ataque LDAP Inyección

Tipo	Ataque
and LDAP	(& (parámetro 1= valor 1) (parámetro 2= valor 2))
or LDAP	(|(parámetro 1= valor 1)(parámetro 2= valor 2))
and Blind LDAP	(&(objectClass=*) (objectClass=recurso)) (&(objectClass=objeto) (type=tipo*))
or Blind LDAP	(|(objectClass=*) (objectClass=recurso)) (|(objectClass=objeto) (type=tipo*))

Para automatizar la explotación de las vulnerabilidades de inyección LDAP existe herramientas como Tamper Data, WebScarab, TamperIE y otros.

Para evitar estos tipos de ataque se debe cumplir con:

- Configuración de controles de seguridad LDAP.
- Validación, filtración y eliminación de valores no permitidos (caracteres sospechosos, expresiones regulares y cadenas maliciosas).

4.4 Contramedidas contra ataque de Inyección

Cuáles son las precauciones por cumplir para evitar, detectar, reducir o amortiguar el ataque de inyección en las aplicaciones web. Estas medidas deben considerar todo el equipo a cargo del proyecto web, no es suficiente implementar un control de seguridad. Debe ser un conjunto de controles ante un posible ataque de inyección, debido a que si un control ha sido vulnerado el siguiente deberá ejecutarse.

4.4.1 Consultas parametrizadas

Él envió de consultas a base de datos en forma directa o concatenada en la programación Web no es recomendable, por ser una de las causas principales de ataque de inyección.

Las interfaces de programación de aplicaciones con acceso a base de datos de los diversos lenguajes de programación proponen trabajar con parámetros a consultas SQL a través de marcadores o enlazando variables para que el usuario no pueda acceder directamente. Tiene que trabajar con parámetros cuando las consultas o procedimientos almacenados solicitan enviar variables, pero no se puede usar parámetros para solicitar palabras claves o identificadores de SQL (nombre tabla o columna).

Envió parametrizada o en .NET

```
cmd.Parameters.Add ( "@clave", OleDbType.VarChar, 16));
cmd.Parameters.Value [ "clave"] = password;
```

Envió parametrizado en php:

```
$cmd->bind_param("ss", $usuario, $clave);
```

```
$stmt->bindParam(':usuario, $usuario, PDO::PARAM_STR, 15);
$stmt->bindParam(':clave', $clave, PDO::PARAM_STR, 15);
```

No se puede parametrizar consultas como:

```
select * from @nombretabla
select * from tabla order by @columna
```

4.4.2 Validación de Ingreso de Datos

Permitir que el usuario ejecute código de inyección se da muchas veces porque no están siendo validados los datos que va ingresando. Se tendrá que validar el tipo de datos, tamaño de variable, rango de datos antes de ser ejecutados. La validación de lista blanca y validación de lista negra son 2 tipos de enfoque de validación de entrada sus características son:

- **Lista Blanca:**

  Comprende los datos a ingresar que están permitidos ser ejecutados; considerando el tipo, limitar la longitud, rango y formato de dato y va a ser evaluado en la lógica de negocio.

  Expresiones regulares en java:

  ```
  Pattern var = Pattern.compile("[^a-zA-Z0-9.@_-~#]+");
  ```

  Expresiones regulares en .net

  ```
  <asp: RegularExpressionValidator id ="nombreREx" runat = "server"
  ```

ControlToValidate = "txtnombre"

ErrorMessage = "Solo letras”

ValidationExpression = "^[A-Za-z] {10,15} $" />

Expresiones regulares en Php

if (! nombre.match ("/ ^ [a-zA-Z] {10,15} $ / D", $usuario)

- **Lista Negra:**

  Comprende los datos que no están permitidos ser ejecutados. Entonces se debe de rechazar la entrada de estos datos por ser malicioso para aplicación.

  Para implementar una lista negra se debe usar expresiones regulares donde se liste los caracteres que no están permitidos, por ejemplo:

  '|% | - |; | / \ * | * \\\ | _ | \ [| @ |

  Se debe implementar sistema de gestión de errores que envíen notificaciones solo al administrador. Por ejemplo, establecer en el archivo de configuración web lo siguiente:

```
<compilation defaultLanguage="vb" debug="false" />
<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
</customErrors>
```

4.4.3 Uso de procedimientos almacenados

Los procedimientos almacenados protegen contra algunas técnicas de inyección y agiliza la respuesta de la base de datos a la aplicación. Los usuarios que accede a la base de datos tienen permiso único de realizar lo que el procedimiento

almacenado indica. Los procedimientos almacenados deben ser ejecutados por usuarios con una cuenta de base de datos con privilegios limitados a manera de “wraper”.

Para estar seguros de que los procedimientos almacenados no tengan problemas con ataques de inyección SQL se debe usar la opción ApexSQL Refactor (en SSMS o Visual Studio), que es una herramienta que formatea y refactoriza código SQL. Haciendo uso de 11 refactorización y 160 opciones de formato. Es gratuito, si desea la herramienta completa se solicita al fabricante.

Para proteger el procedimiento almacenado se realiza los pasos que indican las imágenes:

Figura 35 Paso 1 uso de ApexSQL Refactor

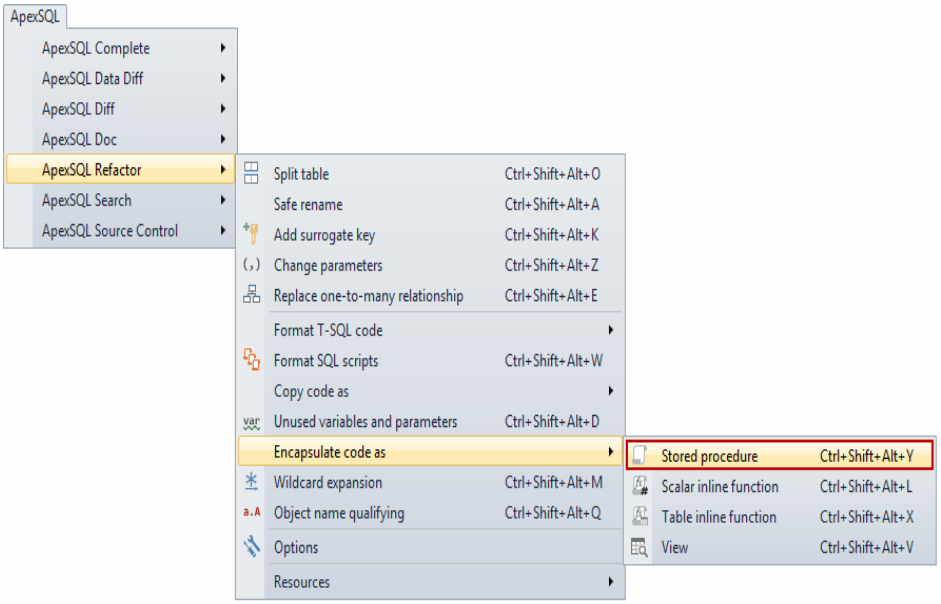
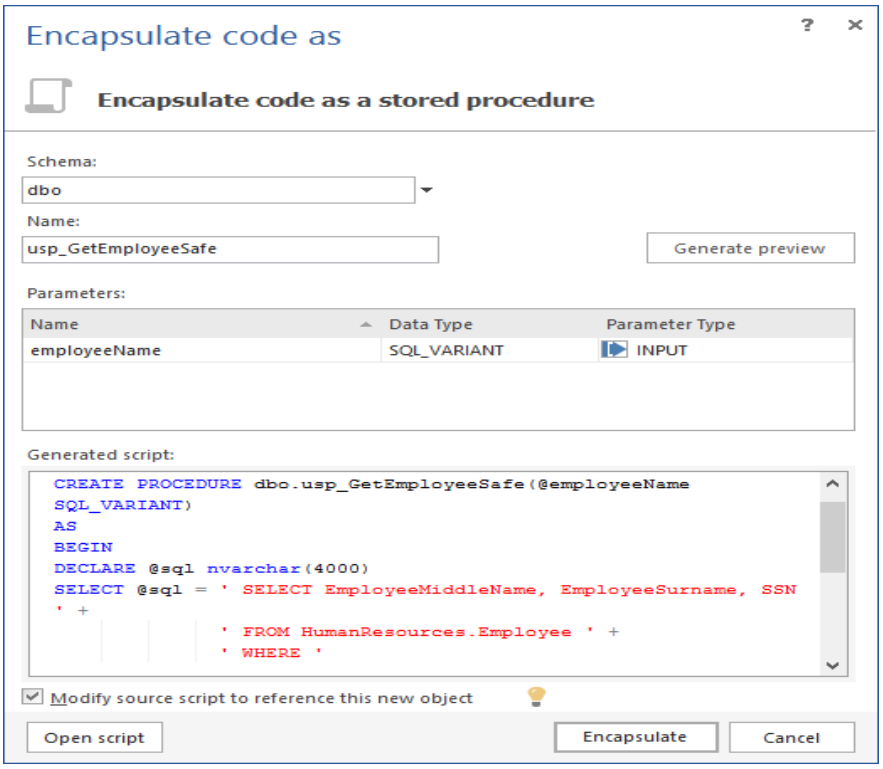


Figura Paso 2 uso de ApexSQL Refactor



4.4.5 Patrones de programación

Para el desarrollo de aplicaciones se recomienda trabajar con patrones de programación que trabajan en capas por ejemplo N-Capas, MVC y otros, lo cual permite la aplicación de cada capa que se abstrae de la concepción global. Siendo estas capas útiles para hacer cumplir que el acceso a datos sea seguro, porque certifica que las consultas a la base de dato sean mediante parámetros. La verificación y validación de los procesos de consultas a datos no solo debe ser evaluado en la capa aplicación o presentación o vista, se tiene evaluar y realizar seguimiento todas las capas con pruebas de inyección.

4.4.6 Gestión de datos:

Una contramedida contra inyección SQL es evaluar y proteger la información confidencial del usuario, organización o base de datos que será manipulado (consultar, registrar, actualizar, eliminar). Se debe considerar las siguientes medidas:

- Información como contraseña, número de tarjeta crédito, archivos y otros se recomienda que sean cifradas con algoritmos que sean seguros. Y no seleccionar algoritmos que son vulnerables y fácil de descifrar.
- Los desarrolladores deben evitar asignar nombres comunes a los objetos críticos como contraseñas, nombre de columnas, funciones, nombre de tablas, etc. Por lo que los atacantes usan diccionarios para realizar ataques automatizados.

4.4.7 Contramedidas en Servidor Base de Datos.

- Los servidores de base de datos por defecto asumen la función pública predeterminada asignándole el rol de administrador. Por lo tanto, tienen el control absoluto del servidor de base de datos. Por lo tanto, se debe configurar permisos personalizados según el tipo de usuario y grupo específico.
- Los usuarios que interactúan con la aplicación; en el servidor de base de datos ellos deben tener únicamente los permisos que la aplicación requiere. Este lineamiento va a mitigar significativamente el riesgo de ataque de inyección SQL
- Muchas aplicaciones no auditan la base de datos, entonces no se podría medir la integridad de los datos si hubo o no ataque de inyección SQL. Se

podría aplicar trigger para controlar las acciones realizadas en la base de datos. También la base de datos tiene que ser actualizada y parcheada permanentemente

- Suprimir mensajes de error ante una manipulación incorrecta por casualidad o adrede de parte del usuario, debido a que los errores muestran información de la base de datos que servirán para ejecutar ataque de inyección. O caso contrario se podría programar una respuesta personalizada para cada tipo de error.
- Desactivar los servicios Web Description Language (WSDL), los servicios web son a menudo tan vulnerable a la inyección SQL como aplicaciones web. Para encontrar vulnerabilidades en los servicios Web, los atacantes tienen que saber cómo comunicarse con el servicio Web, es decir, los protocolos de comunicación compatibles (por ejemplo, SOAP, HTTP GET, etc.), los nombres de métodos y parámetros esperados. Toda esta información se puede extraer desde el archivo de Web Services Description Language (WSDL) del servicio Web. Por lo general, esto se invoca,

  ¿agregando el valor? WSDL al final de la URL del servicio Web. Siempre que sea posible, es una buena idea para suprimir esta información de intrusos no deseados.

  Configuración en .net

  <WebServices>

  <protocolos>

  <Remove name = "documentación" /> </ protocolos>

  </ webServices>

4.4.8 Plataformas de Defensas

Los ataques de inyección no solo se pueden evitar mediante código en la aplicación, también existen herramientas que actuaran durante la ejecución evitando el ataque. Deben ser configurados y evaluados contra técnicas y herramientas de inyección.

4.4.8.1 Cortafuegos de aplicación Web (WAF):

Es un dispositivo presentado en dos modalidades como hardware o software que protege a los servidores web de ataques comunes como SQL Inyección, monitorea el tráfico para detectar alguna anomalía en contra del funcionamiento correcto de la aplicación. El WAF será implementado en el servidor de aplicaciones o la red.

a) ModSecurity:

Es un WAF de código abierto, disponible bajo la licencia GNU. Posee varias funcionalidades como:

- Filtrado de peticiones
- Técnicas anti-evasión (eliminación de barras, decodificación de URL entre otras), Comprensión de HTTP
- Filtrado HTTPS y Análisis Post Payload
- Bloqueo de IP entre otras funcionalidades que protegen la información del aplicativo web.
- url: https://modsecurity.org/

b) OWASP Naxsi:

Es un WAF de código abierto, disponible bajo la licencia GNU, se diferencia de los otros WAF porque cuando detecta reiteradas veces que caracteres inesperados tratando de vulnerar la aplicación, la petición será denegada y el usuario redireccionado una página que le prohíba el acceso.

- Maneja análisis a tiempo real
- Control de tráfico de red
- Prohibir símbolos peligrosos y palabras clave SQL
- url: https://www.owasp.org/index.php/Projects/OWASP_NAXSI_Project

c) NetScaler AppFirewall:

Es un WAF comercial, mantiene la seguridad del aplicativo Web evitando las pérdidas de datos corporativos o clientes. Cumple con los lineamientos de seguridad de la información, como PCI-DSS.

4.4.8.2 Firewall de base de datos

Es un servidor proxy ubicado entre la aplicación y base de datos. Las consultas de la aplicación serán enviados al proxy para que sean evaluados, si no existe ningún indicio de ataque la consulta será enviada al servidor de base de datos. Monitorean y auditan el acceso a la base de datos.

GreenSQL es un cortafuegos de base de datos de código abierto que se puede usar para proteger las bases de datos MySQL y PostGRE SQL. Actúa como un proxy inverso para las conexiones SQL y supervisa todas las conexiones al servidor de la base de datos.

4.4.9 Herramientas Open Source OWASP

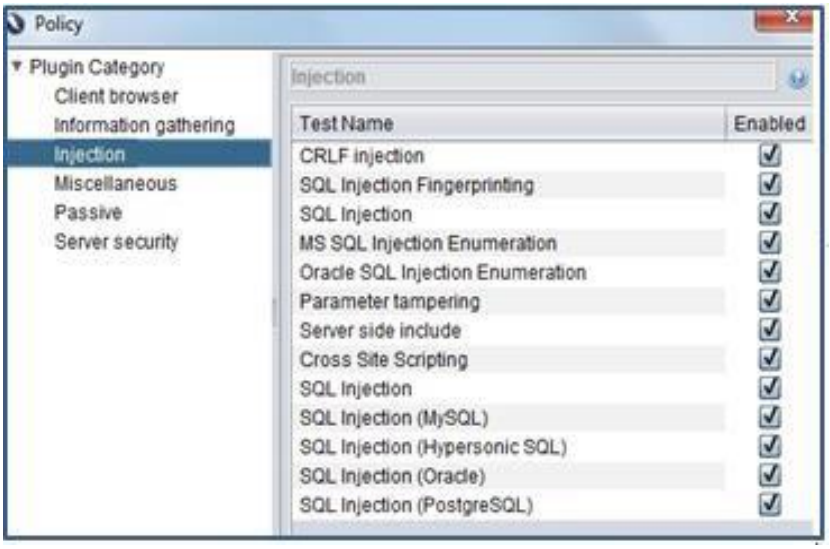
Las empresas que recién inicia y cuentan con recursos básicos, puede evaluar la seguridad de sus aplicaciones haciendo uso de los recursos y herramientas gratuitas que ofrece OWASP que son de alto nivel. Mencionamos las características de algunas de ellas:

a) OWASP Zed Attack Proxy (OWASP ZAP)

Es una herramienta que se puede automatizar la búsqueda de vulnerabilidades durante las etapas de desarrollo y evaluación de la aplicación. También es usado para explotar vulnerabilidades. Herramienta multiplataforma compatible a todo sistema operativo.

- Realizar varios ataques a la vez
- Uso de certificados SSL dinámicos
- Ataque fuerza bruta
- Retorna reporte de vulnerabilidades y contramedidas
- **url: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project**

Figura 36 Configuración ZAP ataque Inyección



b) OWASP Web Testing Environment (WTE)

Conformado por conjunto de herramientas que pueden ser automatizadas para evaluar la seguridad y explotar vulnerabilidades de aplicación Web. Presentado en distribución Linux (Ubuntu) disponible para máquinas virtuales, imágenes ISO e instalaciones basadas en la nube.

url: http://appseclive.org/downloads/

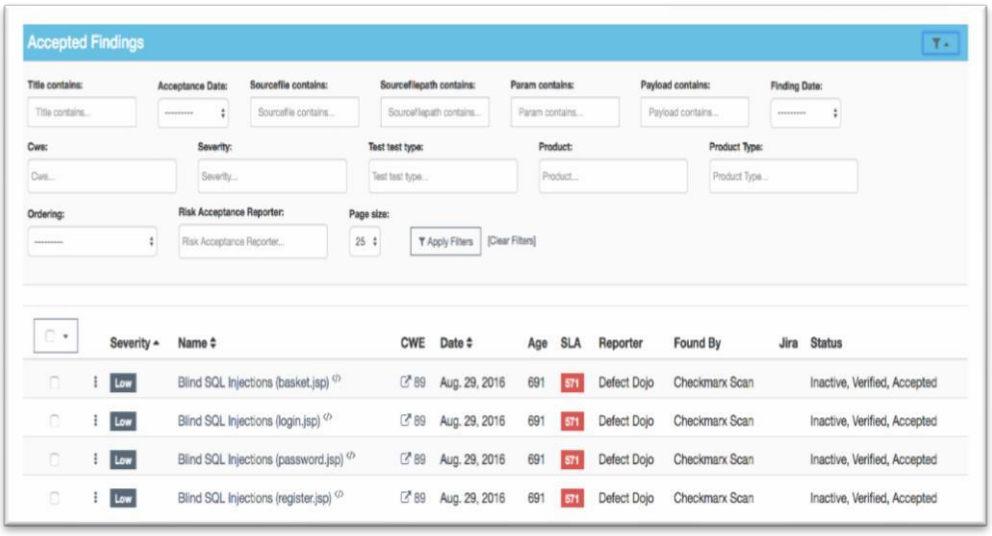
Figura 37 Herramienta de WTE

c) OWASP DefectDojo

Herramienta desarrollada en Phyton, ayuda agilizar y automatizar la gestión de evaluación de vulnerabilidades de aplicaciones web. Los informes y métricas de evaluación son presentados en forma de dashbord (forma dinámica de mostrar la información incluye tablas, gráficos estadísticos y segmentación)

url: https://www.owasp.org/index.php/OWASP_DefectDojo_Project

Figura 38 Resultado Defect Dojo

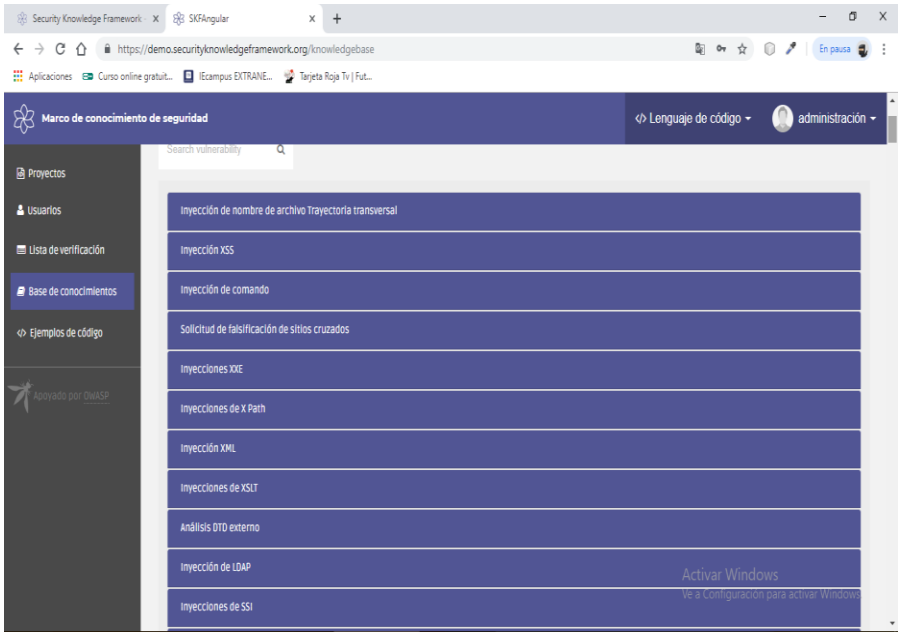


d) OWASP Security Knowledge Framework

Aplicación desarrollada en Phyton; que sirve como modelo a desarrollar, validar y verificar la seguridad de una aplicación. Es recomendable para las realizar capacitación sobre desarrollo de aplicaciones web seguras.

url: https://www.securityknowledgeframework.org/

Figura 39 Marco de conocimiento de seguridad

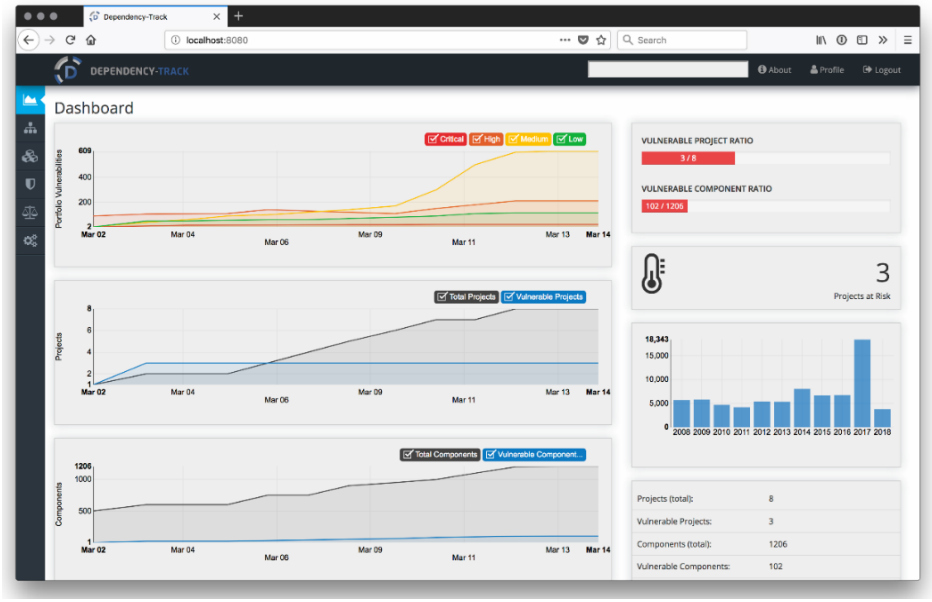


e) OWASP Dependency-Track

Herramienta gratuita que evalúa en forma continua los componentes que son enviados a la aplicación web por terceros. Identifica las vulnerabilidades que presenta el portal web. Presenta el análisis de evaluación en una plataforma inteligente donde las organizaciones puedan tomar decisiones asertivas contra las posibles amenazas.

url: https://dependencytrack.org/

Figura 40 Dependency-Track



CAPÍTULO V

PROPUESTA DE SOLUCIÓN

CONCLUSIONES

1. Los datos estadísticos evidencian el alto riesgo que en la actualidad gran porcentaje de soluciones web están siendo víctimas de los ataques de inyección; por ello se propone que las entidades que desarrollan soluciones web para que logren construir aplicaciones de calidad, primero deben de implementar y cumplir con una política en su organización basados en estándares y modelos de aseguramiento de calidad de software o si es realizado por terceros exigir que cumplan. Además, el modelo de calidad de desarrollo de software se debe amoldar a los requerimientos y recursos de la organización.

2. Para identificar las zonas vulnerables de las soluciones web propenso a los ataques de inyección, los involucrados en desarrollo de solución web deben de realizar de las 2 formas manual o automática. Manual porque el usuario conoce la lógica de la aplicación para buscar las zonas de ataque algo que una herramienta no podría analizar. Y en forma automatizada la herramienta puede ahondar puntos críticos que el desarrollador puede obviar.

3. Según la investigación una vez identificado las zonas vulnerables contra ataque de inyección estas deben ser explotadas y evaluadas para medir el

grado de riesgo. Estos procesos se realizarán en forma automatizada para obtener informes estadísticos y reportes de contramedidas para elegir decisiones asertivas. La eficacia de las herramientas está determinada por el grado de conocimiento, manejo y experiencia del evaluador en aplicaciones web.

4. Se evidencia los errores comunes que suelen cometer los involucrados desarrollar soluciones Web y sus consecuencias; por ello se propone lineamientos de seguridad a cumplir tanto en la programación, implementación de la solución web y configuración correcta de las herramientas que evitaran los ataques de inyección.

RECOMENDACIONES

1. Es inconcebible que el ataque de inyección en la actualidad siga ubicado como primera amenaza de una aplicación web. Se recomienda por lo tanto a las entidades que desarrollan software deben permanentemente capacitar y evaluar a su personal. Actualizar los recursos a usar para desarrollar soluciones web que eviten las nuevas técnicas de ataque de inyección.

2. Los desarrolladores deben considerar que la implementación de framework para la construcción de software es una tendencia en la actualidad, porque contiene funciones programadas contra ataques de inyección. Esto solo resultara si ya no se siguen enviando las consultas en forma dinámica o concatenadas a pesar de que los framework están configurados para enviar variables en forma parametrizada.

3. Los involucrados en desarrollar soluciones web deben de registrar el seguimiento y las contramedidas que se ejecutó ante ataques de inyección en el proceso de desarrollo, implementación y ejecución de la aplicación web. Para posible incidencia se repita nuevamente, saber entonces que hacer.

4. La alta dirección debe estar muy comprometido durante el desarrollo e implementación del proyecto web. Proponer indicadores que permitan evaluar la calidad de la aplicación, realizar auditoria adquiriendo los servicios de terceros que no tengan ningún vínculo con el equipo de desarrollo y que sean reconocidos en el entorno de seguridad.

Bibliografía

Amado Ballen , J. W., Ayala Calderón, C. A., & Sierra Morales , A. A. (2017). *Análisis de vulnerabilidades en aplicaciones Web desarrolladas en PHP Versión 5.6.24 con base de datos MYSQL Versión 5.0.11 a partir de ataques SQL Inyección.* Bucaramanga.

Briones Pincay , G. H., & Hernandez Peñaherrera , E. B. (2018). *Auditoria de Seguridad del Servidor Web de la Empresa Publynext S.A. utilizando Mecanismos Basados en OWASP.* Guayaquil.

Consultores, S. (05 de 2012). *http://sapsiconsultores.blogspot.com/*. Obtenido de http://sapsiconsultores.blogspot.com/2012/06/estadisticas-cyber-ataques-mayo-2012.html

Fuyo Gonzales, J. C., & Rivera Oviedo, J. A. (2016). *Seguridad de JAX-RS frente a ataques por inyección de código.* Bogota.

Hackmageddon. (05 de 2013). *http://hackmageddon.com/*. Obtenido de http://hackmageddon.com/2013-cyber-attacks-statistics/

Hernández Reveles, J. G. (2011). *Identificación y Clasificación de las mejores practicas para evitar inyeccion SQL en aplicaciones Desarrolladas en PHP y PostgreSQL.* Zacatecas.

INDECOPI. (2007). *Norma Técnica Peruana ISO/IEC 17799:2005.*

ISO/IEC 27001. (2005). *Tecnología de la Información - Técnicas de seguridad - Sistemas de gestión de seguridad de la información - Requerimientos. ISO/IEC.*

Jimenez Paneque, R. (1998). *Metodologia de la Investigación.* La Habana: Ciencias Médicas del Centro Nacional de información de Ciencias Médicas.

Tedeschi, N. (s.f.). *Microsoft Developer Network*. Obtenido de https://msdn.microsoft.com/es-es/library/bb972240.aspx

Valverde Chavez, J. A. (2017). *Los riesgos de seguridad de websites y sus efectos en la informacion de medianas empresas de Lima Metropolitana.* Lima.

Verizon. (31 de 12 de 2012). *Verizon.* Recuperado el 31 de 12 de 2012, de www.verizonenterprise.com: http://www.verizonenterprise.com/resources/reports/rp_Informe_Sobre_Investigaciones_de_brechas_2012_es_xg.pdf

ANEXO

Operacionalización de variables

Variable	Definición conceptual	Definición operacional	Dimensión	Indicador	Ítem
Desarrollo de Sistemas Web	Es la construcción de portal web desde la etapa de recolección de requisitos, análisis, codificación, prueba e implementación a un servidor web.	Metodología de análisis de sistemas Metodología de patrón de diseño Metodología de evaluación de funcionalidad de sistema	Funcionalidad	#controles que no ejecutan	4
				# veces de caída de sistema	5,10
			Usabilidad	#cantidad errores codificación	5,10
				# nivel dificultad de manipulación	7
Mejores prácticas contra ataque de inyección	Comprende la elección de norma, modelos y controles de seguridad para asegurar como resultado un producto de calidad.	Estándar de desarrollo de software de calidad Modelo de mejora de calidad de software Buenas prácticas de programación.	Fiabilidad	# cantidad de vulnerabilidades	1,2
			Eficiencia	# cantidad de vulnerabilidades	3, 6,8
			Seguridad	# cantidad de vulnerabilidades	1, 4, 6,9
			Mantenibilidad	# cantidad de vulnerabilidades	5, 8, 10

Matriz de Consistencia

Título: “MEJORES PRÁCTICAS EN DESARROLLO DE SISTEMAS WEB CONTRA ATAQUE DE INYECCIÓN”

PROBLEMA	OBJETIVOS	VARIABLES	METODOLOGÍA
PROBLEMA GENERAL ¿De qué manera los involucrados en desarrollar soluciones informáticas deberán construir Sistemas Web para evitar o contrarrestar los ataques de Inyección? **PROBLEMAS ESPECÍFICOS** 1. ¿Cuáles son las técnicas y/o herramientas que identifique las vulnerabilidades contra ataque de Inyección a Soluciones Web? 2. ¿Qué herramientas usar para explotar y evaluar soluciones web contra ataques de Inyección?	**OBJETIVO GENERAL** Proponer a los involucrados en construir soluciones informáticas, las mejores prácticas de desarrollo contra ataques de Inyección a Sistemas Web . **OBJETIVOS ESPECÍFICOS** 1. Identificar las zonas vulnerables de una aplicación web propenso a los ataques de Inyección a través de técnicas y/o herramientas de forma manual y automatizada. 2. Determinar las herramientas adecuadas para la explotación y evaluación de sistemas web contra ataques de Inyección. 3. Mostrar los lineamientos de seguridad a cumplir para evitar	**Variable 1:** Desarrollo de Sistemas Web **Variable 2:** Mejores prácticas contra ataque de inyección	**Tipo de Investigación** Explicativo **Diseño de Investigación** Experimental **Población y Muestra** **Población:** Personas involucradas en el desarrollo de proyecto web , a los que adquieren el servicio de outsourcing para la construcción de un portal Web , facilitadores y estudiantes de aprendizaje de desarrollo web. **Muestra:** 200 personas de las cuales son 30 profesionales expertos, 50 profesionales junior, 40 docentes y 80 alumnos. **Técnicas** Encuesta Observación

3. ¿Cuáles son los errores típicos que no se debe realizar durante el desarrollo e implementación de aplicaciones web, que al ser ejecutados son propensos a los ataques de Inyección?	los errores típicos propensos a los ataques de Inyección durante el desarrollo e implementación de aplicaciones web.		**Instrumentos** Cuestionario Estructurado Laboratorio de Testing

Encuesta (Cuestionario Estructurado) :

Muestra: 30 profesionales expertos, 50 profesionales junior, 40 docentes y 80 alumnos.

Los resultados de las encuestas fueron la siguiente:

S: Si

N: No

AV: Algunas Veces

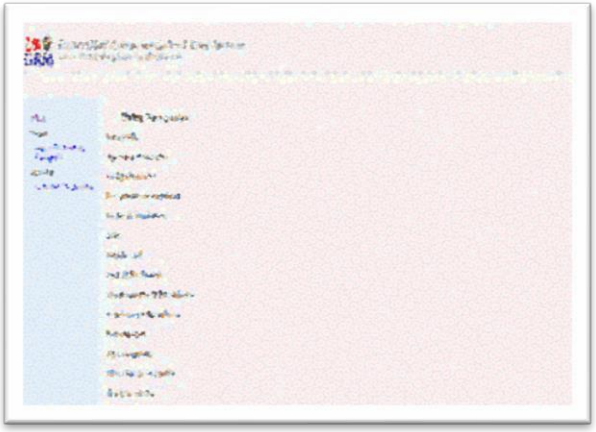
Nº	PREGUNTAS	Experto			Junior			Docente			Alumno			Total		
		S	AV	N	S	AV	N	S	AV	N	S	AV	N	S	AV	N
1	¿Realiza eventos de programación para evitar ataques de inyección?	20	5	5	10	10	30	16	8	16	4	3	73	50	26	124
2	¿Protege las credenciales de los usuarios?	30	0	0	44	3	3	33	4	3	23	15	42	130	22	48
3	¿Toda entrada de datos de parte de usuario es validada?	25	2	3	37	6	7	35	5	0	42	14	24	139	27	34
4	¿Desarrollas sistemas basándose en algún modelo de calidad de software?	19	5	6	7	6	37	5	13	22	4	10	66	35	34	131
5	¿Registra en un historial las incidencias del sistema?	22	3	5	33	4	13	13	9	18	14	8	58	82	24	94
6	¿Considera las buenas prácticas para desarrollar aplicaciones web?	19	5	6	19	12	15	26	8	10	15	13	52	75	39	86
7	¿Evalúa el tiempo de respuesta las peticiones del usuario al sistema?	23	4	3	35	8	7	26	8	6	28	9	43	112	29	59
8	¿Realiza prueba de seguridad durante los ciclos de desarrollo de la solución informática?	21	7	2	29	9	12	6	5	29	12	11	59	68	30	102
9	¿Maneja o se basa en alguna política o estándar de seguridad para desarrollo de Soluciones Informática?	18	8	4	13	7	30	5	8	27	4	5	71	40	28	132
10	¿Antes de implementar su proyecto Web es evaluado/auditado por terceras personas?	19	5	6	25	12	13	20	9	11	9	7	64	73	33	94

Observación (Laboratorio Testing)

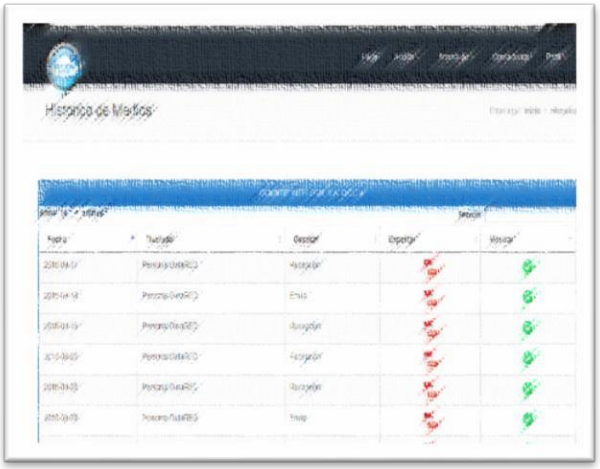
Evidencia de prueba manual y haciendo uso de la herramienta SQLMap, se muestra algunas paginas que tuvieron acceso mediante ataque de Inyección. Algunas organizaciones como:

Pruebas Manual

1.- Institución Académica



2.- Entidades de Salud



Herramienta SQLMap

1.- Tiendas Online

2.- Sistema de Gestión